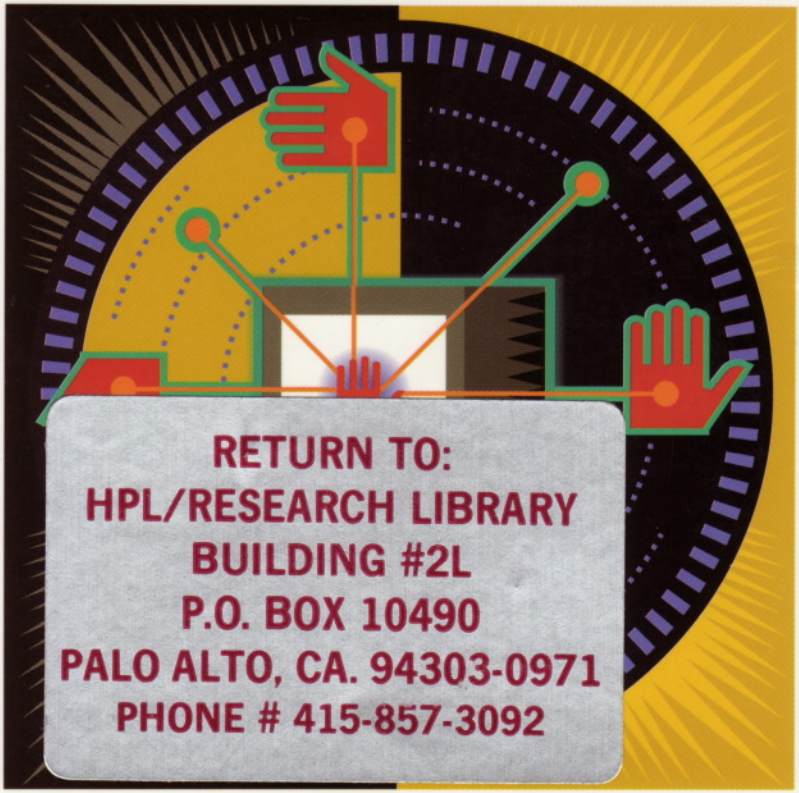
19th Annual HP User Conference and Expo

Interex '93

San Francisco, CA ▪ September 19–23, 1993



VOLUME 2

Proceedings

Sponsored by **Interex,** The International Association of Hewlett-Packard Computer Users

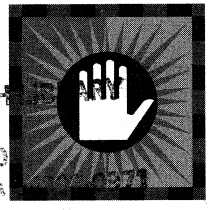
Proceedings

Volume 2 of the

19th Annual HP User Conference and Expo

Interex '93

in
San Francisco, CA
September 19–23, 1993



Interex The International Association of Hewlett-Packard Computer Users

DISC SPACE: HOW MUCH IS ENOUGH?

by Vladimir Volokh
VESOFT
1135 S. Beverly Dr.
Los Angeles, CA 90035 USA
310-282-0420

In spite of many new developments in disc technology, the good old Winchester drive, with its mechanically movable arms, is still the primary medium for all of our files.

In this article I will try to present some observations on HP3000 file structure -- both for "Classic" (MPE/V) and "Spectrum" (MPE/iX) computers -- in the hope that it might help HP3000 users manage disc space better. It seems that many simple questions have answers that are not so simple.

HOW DO WE MEASURE DISC SPACE?

In various discussions about disc space, you've seen terms like "sectors", "kilobytes", "megabytes", and "gigabytes". What do these words mean? Well, nothing is simple. A sector, by HP's definition, is 256 bytes; "kilo" (K) is 1000, or 1024 for memory devices; "mega" is 1000000, or 1024K for memory devices; and "giga" is a prefix denoting a billion, or 1.073 billion for memory devices. My dictionary tells me that "tera" means one trillion (10^12) of a given unit. HP's Glossary of Terms mentions only "kilo" and "mega" (in 1989). So, considering all this mathematics, how many megabytes does your disc have if after :DISCFREE C on your XL machine you see the following:

```
ALL MEASUREMENTS ARE IN SECTORS.
ALL PERCENTAGES ARE RELATIVE TO THE DEVICE SIZE.

           |    Configured     |      In Use       |     Available     |
-----------+-------------------+-------------------+-------------------+

LDEV :     1 -- (MPEXL_SYSTEM_VOLUME_SET:MEMBER1)
 Device    |    2232192        |    1708144 ( 77%) |     524048 ( 23%) |
 Permanent |    1852720 ( 83%) |    1605344 ( 72%) |     247376 ( 11%) |
 Transient |    1852720 ( 83%) |     102800 (  5%) |     524048 ( 23%) |
```

Considering that 4*256 is close enough to 1000 you can do it easily -- just divide the number of sectors by 4000 and you will have it in megs. In this case, it'll be 2232192/4000 = 558, close enough to the real answer, 545 megs.

HOW BIG IS THE DISC?

As you've seen above, MPE/iX gives you an answer via :DISCFREE. In MPE/V the utility FREE5.PUB.SYS -- true to its name -- shows only free space. But if it shows you that X sectors are free, that's X sectors out of how many? This information is hidden deep inside the VINIT utility (as if it were unimportant). Try this:

```
:VINIT
>pfspace 1;addr
```

... You will see a lot of information about addresses and sizes of free space (and you don't care much about that). But at the end of this listing you will see:

```
TOTAL VOLUME CAPACITY:   216832 SECTORS
TOTAL FREE SPACE AVAILABLE: 16490 SPACE
MAXIMUM CONTIGUOUS AREA: 5505 SECTORS
```

By the way, it's not my typo (if you're wondering about "16490 SPACE") -- it's an unknown MPE designer's mistake, frozen in time ....

HOW MUCH OF IT IS FREE?

MPE/iX gives a pretty straightforward answer to this question: look at its output in the example above -- this time not on the first line but on the second:

```
Permanent |    1852720 ( 83%) |    1605344 ( 72%) |     247376 ( 11%) |
```

As you see, 83% of the whole space is configured to be used as permanent, 72% is used, so only 11% (which is 83-72) is available for permanent files. But why doesn't this simple calculation work for the third line (transient space)?

```
Transient |    1852720 ( 83%) |     102800 (  5%) |     524048 ( 23%) |
```

Even though transient space can also take up to 83% of the space on LDEV 1, in this case only 28% is left for that: 17% can't be permanent and 11% is unused by permanent files; because 5% is actually used by transient space, 23% is available.

On MPE/V machines available space is supposed to be shown by the FREE5.PUB.SYS utility or via the PFSPACE command of :VINIT (16490 sectors in the PFSPACE example above). But what about virtual (transient) space? This information, again, is hidden -- this time inside the :SYSDUMP output:

```
:SYSDUMP $NULL
ANY CHANGES? YES
 ...
DISC ALLOCATION CHANGES? YES
VIRTUAL MEMORY CHANGES? YES
LIST VIRTUAL MEMORY DEVICE ALLOCATION? YES
VOLUME NAME   LDEV #   VM ALLOCATION
   LDEV1        1          25
 ...
ENTER VOLUME NAME , SIZE IN KILOSECTORS (MAX = 255 )?
```

[???] This means that MPE/V knows nothing about virtual space utilization at the moment; some space is also taken (possibly) by spool files and by temporary files. Note also that even though total, free and virtual space is given by DEV#, used space is not. (The :REPORT command gives used filespace-sectors by group and accounts.) One way to know this distribution is to use the MPEX command %LISTF @.@.@,DISCUSE.

IS FREE SPACE REALLY AVAILABLE TO US?

Seeing the FREE5 output on "Classic" one should pay attention not only to the "TOTAL FREE SPACE" line but also to the preceding ones:

```
:RUN FREE5.PUB.SYS
VOLUME MH7945U1            LDEV 1
LARGEST FREE AREA= 25530
  SIZE  COUNT  SPACE   AVERAGE
>100000 0      0       0
>10000  2      42796   21398
>1000   0      0       0
>100    4      540     135
>10     29     1065    36
>1      107    217     2
TOTAL FREE SPACE=44618
```

If you have a lot of small pieces, they might not be usable at all because none of your files may have small enough extents (more on this later). What you need is not just free space but CONTIGUOUS space. On "Classics", disc space can be condensed to some degree by the >COND command in :VINIT; on "Spectrum" machines the disc fragmentation shouldn't be a problem (or so HP tells us).

IS THE "USED" SPACE REALLY USED BY US?

OK, by subtracting "free" space from the "total" space or just looking at the :DISCFREE output we might get an idea of how many sectors are "used" -- physically, that is. Keep in mind, however, that probably about half of those files which you see on the full backup listing **have not been used** (either modified or accessed) for a long time -- 6 months or more. But which half? Some answers to this question can be found in the :STORE command of MPE or, better yet, using selection by ACCDATE and/or MODDATE in MPEX (with totals of files and space). Archiving and purging seldom used files saves a lot of disc space, directory space, and backup time.

TO BLOCK OR NOT TO BLOCK?

Another question is: how is the space used inside "active" files? One factor -- relevant on MPE/V machines but not on MPE/iX machines -- is blocking. MPE/V does all disc I/Os in multiples of one sector (256 bytes). The blocking factor is the number of records that we choose to fit into a certain number of sectors (block). But very often we don't choose -- we simply rely on MPE/V defaults, which can range from good to very bad (see [1] for more details). A bad blocking factor wastes not only disc space, but also I/O time -- the more records per one I/O we read/write, the better. Consider some examples:

```
ACCOUNT=  SYS          GROUP=  OPERATOR
FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

REPORT1         132B  FA          26      10000   1     1251  1  8
REPORT2         132B  FA          26      10000  60      651  1  8
REPORT3         132B  FA          26         26  60       62  1  1
REPORT4         132B  FA          26         26   9       20  1  1
```

Here the file REPORT1 is built with the default blocking factor 1 (1 = 256 bytes / 132 bytes); the remainder (256 - 132 = 124 bytes) is simply wasted, though it's almost 50% of the space; this file is like a piece of swiss cheese -- with many big holes inside. The second file is the result of changing the blocking factor to 60, thus achieving the BEST space utilization for this file -- now 60 records take 60*132=7920 bytes which is close to the size of a block of 31 sectors (256*31=7936). However, we can get an even bigger saving by SQUEEZEing this file (setting FLIMIT down to EOF) -- that's how we got the file REPORT3. By reblocking it again we save more space; as a result, the difference in size between REPORT1 and REPORT4 files is quite significant. Things like this can be done using our very own MPEX (the %ALTFILE command with options SQUEEZE and BLKFACT=BEST).

And what about XL (or should we say iX) computers? The blocking factor does not mean much there; all the records are tightly packed, except for the last extent which can (for very big files) be up to 2048 sectors. The good news is that the FCLOSE intrinsic (on the XL) has a new option called "XLTRIM", which allows the system to reuse free space beyond the end of file without decreasing the file limit. Look at the following before-and-after example:

```
ACCOUNT=  SYS          GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

REPORT1         132B  FA          26      10000   1      256  1  *
REPORT2         132B  FA          26      10000   1       16  1  8
```

Quite a savings (MPEX's %ALTFILE ;XLTRIM does it) -- and we can append to the file!

THE EXTENT QUESTION, OR WHERE THE FILE IS?

The extent is MPE's compromise between two extremes in file size management: assigning all file space requested to the file immediately or giving space one record (or sector) at a time. In MPE/V a file can consist of anywhere from 1 to 32 extents (the default number is 8). Each extent resides wholly on one disc, but different extents may be located on different discs. So where is any given file? You have to know the full extent map of the file and only then can you think about improving system performance through disc balancing. If you use the LISTDIR5 >LISTF you might see the DISC DEV # line, but this of course is only the device of the first extent (the same goes for :STORE listings). >LISTF ...;MAP, however, gives you a map (the first digit is the "volume table index", which is not necessarily the device number, and is hard to convert to the device number):

```
LISTDIR5 G.06.00 (C) HEWLETT-PACKARD CO., 1983
>LISTF VESOFT.PUB.SYS

FCODE: 0              FOPTIONS: STD,ASCII,VARIABLE
BLK FACTOR: 1         CREATOR: **
REC SIZE: 1276(B)     LOCKWORD: **
BLK SIZE: 640(W)      SECURITY--READ:    ANY
EXT SIZE: 10(S)                WRITE:   ANY
# REC: 482                    APPEND:  ANY
# SEC: 70                     LOCK:    ANY
# EXT: 7                      EXECUTE: ANY
MAX REC: 13                 **SECURITY IS ON
MAX EXT: 7            COLD LOAD ID: %24025
# LABELS: 0           CREATED: THU,  9 APR 1992
MAX LABELS: 0         MODIFIED: THU,  9 APR 1992
DISC DEV #: 3         ACCESSED: THU,  9 APR 1992
DISC TYPE: 3          LABEL ADR: **
DISC SUBTYPE: 4       SEC OFFSET: %5
CLASS: DISC           FLAGS: NO ACCESSORS

>LISTF VESOFT.PUB.SYS;MAP

FCODE: 0              FOPTIONS: STD,ASCII,VARIABLE
BLK FACTOR: 1         CREATOR: **
REC SIZE: 1276(B)     LOCKWORD: **
BLK SIZE: 640(W)      SECURITY--READ:    ANY
EXT SIZE: 10(S)                WRITE:   ANY
# REC: 482                    APPEND:  ANY
# SEC: 70                     LOCK:    ANY
# EXT: 7                      EXECUTE: ANY
MAX REC: 13                 **SECURITY IS ON
MAX EXT: 7            COLD LOAD ID: %24025
# LABELS: 0           CREATED: THU,  9 APR 1992
MAX LABELS: 0         MODIFIED: THU,  9 APR 1992
DISC DEV #: 3         ACCESSED: THU,  9 APR 1992
DISC TYPE: 3          LABEL ADR: **
DISC SUBTYPE: 4       SEC OFFSET: %5
CLASS: DISC           FLAGS: NO ACCESSORS
EXT MAP: %300161067   %200233735   %300162124   %200240307
         %300162207   %100211521   %200240326
>
```

In MPE/XL file labels are kept separately from the data, and yet :LISTF ,3 still shows the file label address, which might have no relevance to the location of the data at all. Here is an example of :LISTF ,3 and MPEX's %LISTF ,4 showing the full extent map of the file:

```
:LISTF LOG3320,3

FILE CODE : 0                   FOPTIONS: BINARY,VARIABLE,NOCCTL,STD
BLK FACTOR: 1                   CREATOR : **
REC SIZE: 2044(BYTES)           LOCKWORD: **
BLK SIZE: 2048(BYTES)           SECURITY--READ    : CR
EXT SIZE: 0(SECT)                         WRITE   : CR
NUM REC: 2720                             APPEND  : CR
NUM SEC: 2304                             LOCK    : CR
NUM EXT: 9                                EXECUTE : CR
MAX REC: 1024                           **SECURITY IS ON
                                FLAGS   : 1 ACCESSORS,SHARED,1 R,1 W
NUM LABELS: 0                   CREATED : THU, APR  9, 1992,  2:01 PM
MAX LABELS: 0                   MODIFIED: THU, APR  9, 1992,  2:01 PM
DISC DEV #: 1                   ACCESSED: THU, APR  9, 1992,  2:01 PM
SEC OFFSET: 0                   LABEL ADDR: **
VOLCLASS  : MPEXL_SYSTEM_VOLUME_SET:DISC

                        MPEX %LISTF log3320   PAGE 1
           MANAGER.SYS,PUB   THU, APR  9, 1992,  4:01 PM

ACCOUNT=  SYS         GROUP=  PUB

-----FILE------ EXTENTS          -----SECTORS-----  DEVICE
NAME      CODE  NUM MAX         USED NOW  SAVABLE  CLASS

LOG3320         10   *              2560      208  DISC
   Dev/Sector:    2/%0000004516700   2/%0000000444440   2/%0000000072040
   Dev/Sector:    3/%0000001407620   3/%0000007737620   1/%0000003207300
   Dev/Sector:    1/%0000002675520   2/%0000006572620   3/%0000000536200
   Dev/Sector:    2/%0000006577760
```

To finish this little essay I propose a puzzle to MPE/iX users: what do these two "*" mean in the following :LISTF ,2 ??

```
ACCOUNT=  SYS         GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

PUZZLE           128W  FB        1608       2222   1     1616  *  *
```

The answer is in one of the recommended reading items:

1. Eugene Volokh, "The Truth About Disc Files", Presented at 1982 HPIUG Conference, San Antonio, TX USA

2. Andy Tauber, "Disc Balancing", INTERACT Magazine, Jan. 1986

3. Greg Englestad, "HP3000 Disc Management", Supergroup Association Magazine, Sep.-Nov. 1987

4. Eugene Volokh, "The Truth About MPE/XL Disc Files", Presented at 1989 INTEREX Conference, San Francisco, CA USA

5. S. Gordon, V. Volokh, "The Art And Science Of Disc Space Management", INTERACT Magazine, July 1991

Paper #5033

HP 3000 to PC Portability and Interoperability

N. M. Demos
Performance Software Group
12 Hillview Drive
Baltimore, Maryland 21228-2237
(410) 242-6777

INTRODUCTION

The context in which the issues of interoperability and portability exist is the current push for "Open Systems". "Open Systems are hardware, software and network environments that are designed and implemented in accordance with standards that are vendor independent, and that are commonly available."(1) Issues such as when is a proposed standard a real standard and what is "vendor independent" are beyond the scope of this paper. It is clear, however, that there is a clear need and a desire for the hardware, software and network products of many different vendors to work smoothly together in such a way that the replacement of any one of the above can be done easily and economically. Everyone may agree that there is a need for Open Systems, but it is hard to find someone who does not see software conversions to new platforms as a diversion from the more productive tasks of an information systems department, if not a waste of time. While the push for Open Systems has come from the government and larger businesses the desirability of Open Systems is not lost on the smaller user. Would it not be wonderful if a user could order any software with the assurance that it would run on his current computer system with minimum effort on his part? One reason that the IBM PC and its clones will be with us for the foreseeable future is the huge amount of packaged software that is available for this platform. As an operating environment the IBM PC architecture is obsolete and presents a developers' nightmare, but only when a new architecture is widely available at a reasonable cost will this architecture die. We hope to be seeing this with IBM OS/2 and especially, with the introduction of Windows NT. Time will tell if this is the case.

In the Open Systems environment there are several terms that need definition before we start:

Portability - The ability to run on dissimilar computers without rewriting code(1).

Scalability - The degree that hardware or software "has the ability to run on platforms ranging from a small desktop to a mainframe host without rewriting"(1).

Interoperability - The ability to share data (without conversion) throughout an enterprise and to communicate effectively, efficiently and instantly inside and outside the enterprise(1).

Migration - To move software from one system (hardware or operating system) to another.

Compatibility - In data processing, the capability of a computer system, through special hardware and/or software, to run the object programs of another computer system. Compatibility usually imposes a significant performance penalty and is viewed as a temporary assistance in migration.

UNIX - An operating system considered the best Open Systems choice, in spite of the fact that it is technically a proprietary system of AT&T. UNIX has been almost universally accepted in the academic and scientific world as the standard operating system and it is in the process of being accepted in business computing also. However, it is not as "standard" as one would desire. In spite of the fact that there are many variations, it at least is by intent and design meant to be a portable operating system. In summary, it is far from ideal, but the best that there is.

POSIX (Portable Operating System Interface X) - The most important UNIX based standard. There are currently POSIX standards for system calls, commands, utilities and systems administration.

Client-Server - A term used to describe a system of distributed computing where multiple computers (Clients) request information, service or resources from one or more other computers (Servers). In the Client/Server model the Client typically executes the program and therefore has control, relying on the Server for data base files and other resources.

(Seamless) Integration - (Integrate - "to form or blend into a whole" [Webster's Ninth New Collegiate Dictionary]). In particular reference to the Client/Server model, the coupling of the computers and software. The ultimate goal is "seamlessness", i. e. the user does not see the "seams" and is not aware that he is accessing more than one system.

Legacy System - (Legacy - "something received from an ancestor or predecessor or from the past" [Webster's Ninth New Collegiate Dictionary]). An application system that is currently operating and is necessary to the continuing activity of the business entity. Legacy systems, which exist in almost every situation where

movement to a new hardware/software system is contemplated, vastly complicate the process and may severely limit the options.

Portability and interoperability are closely related. If one can run the identical software system in the same environment on two or more platforms, then the systems are both "portable" and "interoperable", assuming there is some communications infrastructure in place to allow the platforms to communicate. However, all interoperable systems are not portable. Data may be able to be freely exchanged among platforms, but the software may not run on all platforms.

OPEN SYSTEMS AND STANDARDS

There is no doubt about the desirability of Opens Systems and standards. Much literature has been directed to the computer community about the desirability and applicability of Open Systems and standards. Unfortunately, the availability of workable Open Systems in the commercial arena is another story. Even UNIX is not a truly "Open System"; it is proprietary to AT&T. Defining and implementing an "Open System" in the dynamic environment the computer industry now finds itself may be an impossible task. Of greater importance is the inadequacy of "standard" UNIX for business applications. For example, the file system seems archaic to one used to the file and I/O structure available on the HP 3000. Security is another area in which UNIX is weak. A true batch facility is not available, although programs can be run in the background. The examples go on and on.

To overcome some of the more severe limitations, hardware vendors have added extensions to their versions of UNIX. In one analysis (2), it was determined that the POSIX standards covered only a small proportion of the HP-UX calls. While some of these calls may be incorporated in the standards at a later date, it becomes obvious that we are a long way from an acceptable commercial Open System. The choice for the system implementor seems to be standardization at the cost of functionality or vice versa.

In other areas standardization faces the same problems. The standards are way behind system and application requirements, so the use of extensions becomes inevitable. A good example of the failure of standards to meet business requirements is COBOL. COBOL was the first big attempt at standardization in the commercial information systems area. After an initial reluctance on the part of the programming community, it was generally accepted. However, the standards did not keep up with the need for screen handling and data base access. Therefore, programmers had to add system calls to COBOL programs to be able to work with a screen handler and/or a data base. We now have millions of lines of

COBOL code that are theoretically portable but they are really not because of systems calls to add required functionality not available in the standard! COBOL is available on many platforms, but what is the percent of HP 3000 software that could be easily ported to these platforms? Since most HP 3000 COBOL programs use either Image or Vplus, it is very low.

The current inadequacy of UNIX and standards does not mean that the system implementor should disregard UNIX and other standards. Even if a program or system cannot be implemented in a fully "standard" way, the advantages of coming as close as reasonably feasible are many. For example, the use of standard methodology has a positive impact on the training required to make a new employee productive. Using standard methodology where feasible is usually better than using proprietary systems because in today's environment manufacturers are much more inclined to enhance their "standard" products than their proprietary ones. There are still significant benefits in portability and interoperability even though the use of standards in the implementation of any given system or application may not be complete. The above examples and comments on the failure and/or inadequacy of standards are not meant to discourage anyone from adhering to standards, but rather to help set reasonable expectations.

ESTABLISHING SYSTEM OBJECTIVES

As in all projects, one of the most important keys to success is the establishment and complete definition of system objectives. Our objective was to port HP's Transact (compiler) to the IBM PC and other platforms. The intended users of our system would employ it for one or more of the following purposes:

1. As a desktop workbench to design, write and test HP 3000 programs. This would remove a load from the production HP 3000 and allow the programmer to design and test at home or while traveling. System objectives for this application include:

    a. Exact functionality between the PC and HP 3000 is required. Any deviation must be precisely documented.

    b. The ability to download and upload test files is necessary.

    c. The dictionary (Dictionary/V) must be up-loadable and down-loadable and the dictionary manipulation and maintenance facilities must be available on the PC.

d. Compile time speed is desirable, but execution speed is not a requirement.

e. Symbolic debugging (in this case Trandebug) must be available and fully implemented on the PC.

f. Any additional facilities available from the Windows environment that make the programming and testing easier are desirable. Examples include better editors than on the 3000 and the ability to edit, compile, debug and see the results (particularly Vplus) in different Windows.

2. As an aid in migration from the HP 3000. In the case of both in house programs and purchased software, there is often a need to do production on other platforms for a variety of reasons. Essential components include:

a. Similar functionality is required, although it is not as stringent a requirement as in 1. above.

b. The ability to download and convert source, flat, data base and Vplus files is necessary. The ability to convert to other data bases and screen handlers is desirable but not essential.

c. Downloading and converting the dictionary is necessary.

d. Compile time speed is not a factor but speed of execution is.

e. The ability to maintain and operate the application in a stand-alone environment in the absence of a HP 3000 is essential.

3. To gain flexibility and similar functionality on multiple platforms. This is similar, but not the same as 2. above. In the case of 2., we are looking at a migration; there will be no need in the future to operate and exchange data with multiple systems. The intent is to allow the same program and data to be used on any one of a multiple number of machines interchangeably. The requirements for this approach are a combination of 1. and 2. above, although the programmer workbench attributes are not as essential, but desirable. An additional important and not trivial requirement exists in this scenario: the ability to access data (particularly data bases) across platforms, in a fast, transparent ("seamless") manner.

DEFINING THE SYSTEM ENVIRONMENT

As we completed our system objectives, it became obvious that establishing the environment in which the system would operate was very important. How much of the environment of the source system has to be transferred to the new system? If translation and re-formatting are required, what parts of this should take place on the source system and how much on the target system? These are very important questions, particularly if one is looking for a generalized solution, rather than one that will be applied just to one software system. In the case we are reviewing here we needed the following elements of the HP 3000 environment on the PC:

1. **Image**. Image is the pervasive data base system on the HP 3000. We considered two ways of solving the problem of Image data bases. We could have used an available data base on the PC and translated Image system calls to the format required and converted the Image data bases to the data base system chosen. This has the advantage of utilizing existing, proven software and providing an additional functionality to the user. The disadvantages are the improbability of being able to do an exact functional duplication with a different data base system in addition to the difficulties in converting the structure and data from Image to another data base. Secondly, there were two vendors who had developed Image simulations on the PC, but they were not functional with Windows. Since we had a stringent requirement to duplicate functionality exactly and wanted our system to work under Windows, we chose to implement a data base structure on the PC that was functionally an exact duplicate of Image on the HP 3000. This also meant that we had to generate software to provide additional functionality such as found in HP Dbschema and Dbutil. Image is not that hard of a system to duplicate. The structure is well documented. Besides the problems with the supporting software, the largest problem is maintaining the integrity of the data base. We implemented simple recovery procedures for use with our current single-user system. As we enhance to a multi-user system, we will have to implement the Image locking strategy and a more sophisticated recovery system.

2. **Vplus**. The Vplus situation is similar to Image, but Vplus is a much more complicated system. The internals of Vplus are not documented and Vplus has grown incrementally as HP released new block-mode terminals and employed new communications technology. We also had the decision of implementing our own version and using someone else's. There were two contenders for Vplus. One is Mistral, a simulation of Vplus on the PC and Software Research Northwest's Wingspan, which is not a simulation, but has a conversion routine to get from Vplus to Wingspan. Mistral is a fairly old implementation (it was first done for the 150!). It is currently not functional with Windows. Wingspan would have given us more functionality than Vplus, but our development team decided that it is more complicated and might not be completely functionally identical. Again, it was

decided to do our own version of a screen handler with identical functionality as Vplus.

3. **HP 3000 intrinsics**. The system (Transact) that we were porting did not use the HP 3000 intrinsics directly. We could have gone directly from Transact to the necessary code to give the same capability. This presented two problems. To get an exact match in execution would have been difficult and the system would not have been nearly as flexible. Therefore, we decided to implement all the Vplus, Image and necessary file handling intrinsics in our system. We would use the same approach as Transact, calling the appropriate intrinsics as required. We also implemented most other intrinsics that we thought would be desirable.

4. **The MPE command interpreter**. We users of MPE forget sometimes how much the MPE command interpreter does for us. For example, we all use file equations. There is no such facility in M-S DOS, although the SET command can be used, it is not the same and does not provide the same level of flexibility as the MPE File command. For that matter, the whole file structure of MS-DOS is different. There is no such thing as record lengths as far as DOS is concerned. Files are merely continuous streams of bytes, possibly with delimiters (CR is used most). To be able to port and use MPE files on the PC, we have to preserve the MPE label information. We did this in the most conventional way; we placed the MPE label information in the beginning of the PC file. It became obvious as the project started that it would be difficult to supply full program compatibility without some more capability to tie the system together. To handle this situation, we wrote our own command interpreter, including support for all commands that made sense on the PC. As a result, we have an environment that is comfortable to the MPE user and programmer. We have also the functionality that programs written for the 3000 could not do without.

METHODOLOGY

Once we had established the systems objectives and defined the systems environment, we could look at the PC software and tools available to accomplish the task. The systems choices on the PC are:

1. **MS-DOS**. MS-DOS is by far the most widely used operating system in the IBM-compatible PC arena. It is an obvious choice. However, DOS alone is becoming obsolete. Most users want a more powerful user interface.

2. **MS-DOS with Windows.** As we looked at the systems available, Windows was maturing. With the advent of Windows, version 3.0 and then 3.1, it was decided that our system should operate in this environment. The memory handling and multiple window capabilities this makes available to us made the task significantly easier.

3. **IBM O/S 2.** One of our goals was portability, so we have designed the system to be easily portable. Although we have not yet tried to port the system to OS/2, we do not see this as a major task.

Portability was an important objective of the software we were developing. We know that later we will be porting this system to UNIX. To make the system as easily portable as possible, all the unique coding required to support the PC (and any other platform) is isolated. Another factor made the goal of portability easier to implement. The development team is used to doing its coding on Apple Macintoshes. It was decided to code the system first on the Mac and then port it to the PC as we went. This forced the developers to isolate the code unique to each platform from the beginning. The capability to run on the Mac, as well as the PC, is a side benefit of this approach.

The choice of programming languages quickly came down to Pascal, C or a combination of the two. Although the developers were more familiar with Pascal, C was chosen for the following reasons:

1. It is more standard than Pascal in its different implementations. Although there is a standard Pascal, it is not sufficient for most tasks, so extensions have been implemented to enhance the language. Unfortunately, they are different on different platforms.

2. C is richer (has more capability) than Pascal.

3. C is rapidly becoming the standard language for systems development (as opposed to application development).

The system was originally coded in standard C. As the project progressed, it was decided that C++ had significant advantages that were more important than the fact that C++ is not as standard as C. The code was changed to C++. Incidentally, this is only one of many changes that were made during the course of development. The software was torn apart and rebuilt a number of times as various problems arose or the code became unwieldy. We viewed this as a necessary burden imposed by the nature of our development effort as well as the continuing evolution of C compilers and operating systems.

TESTING

There were attempts to test the system as it was being built. Unfortunately, this was a time consuming process because the I/O system was one of the later parts of the project. Still, significant information was gained from early testing that was of benefit to the project as it progressed. The testing strategy was as follows:

1. Each individual part was tested by the coder.

2. Test programs were compiled and run. If either one blew up, the reason was isolated.

3. Results were checked for obvious errors.

4. The results were match against the identical programs and data run on the HP 3000.

5. Test scripts were developed so that testing could be conducted in a more organized fashion and tests could be repeated easily to be sure that changes had not impacted older code. These scripts will be invaluable when the system is ported to other platforms.

RESULTS

We now have a system that meets our objectives, although testing and development are still continuing. Some observations from the results are:

1. Porting and interoperability of software are possible and feasible.

2. Attainment of identical functionality is possible only if strict rules are observed and the temptation to add capability as the software is coded is rigorously avoided.

3. Desk top systems are viable targets for porting systems from large computers.

4. Porting the necessary core software is only a small part of the task. Porting the necessary software environment and supporting functions are a major, if not the majority of the task.

5. A complete port of the environment, done in a way that keeps it distinct from the other software has major advantages. For example, we are investigating adding

a COBOL compiler to our system. This will allow us to market to a much wider audience. We are very happy with this result.

References:

1. Umarji, Sudhi R., "Open Systems Environment and Technology," presentation sponsored by Hewlett-Packard, January 1993.

2. Vallefont, P. and Lamant, C., "Portability Needs Under HP-UX," Proceedings of the Interex Conference, August 1992.

3. Johnson, Rod, "OSF: Open Systems Through an Open Process," Proceedings of the Interex Conference, August 1991.

4. Johnson, Rod, "Applications Neutral Distribution Format for Application Portability, and Open Systems," Proceedings of the Interex Conference, August 1991.

5. Ferguson, George, "Migrating to Client/Server," Proceedings of the Interex Conference, August 1991.

6. Barrat, Michael L., "UNIX for the MPE Programmer," Proceedings of the Interex Conference, August 1991.

7. Hirsch, Bernard S., "Using Standard Tools to Build an Open, Client/Server Prototype," Proceedings of the Interex Conference, August 1992.

8. Saylor, Todd and Johnson, Aaron, "Transporting an MPE Application to UNIX," Proceedings of the Interex Conference, August 1992.

5034
Working with Native Mode Spooling

Royden Somerville
Room 114 Computing Center
University of Notre Dame
Notre Dame, Indiana 46556
(219) 631-0581

Hewlett-Packard (HP) has implemented a new version of spooling for MPE/XL and MPE/iX on its HP 3000 900 series computers. Since the new spooling is designed for the Native Mode (NM) environment, it is referred to as Native Mode Spooling. There are significant ways in which Native Mode spooling differs from the older spooling. (This is known on MPE/V simply as spooling, and on MPE/XL as compatability mode (CM) spooling. There are added options in NM Spooling which in some cases offer more control, and which in other cases involve doing things somewhat differently than you may be accustomed to if you use the older spooling. We shall examine the new method with the intention of gaining a better understanding of spooling in general and of HP Native Mode spooling in particular. The approach will be to compare NM Spooling with the previous method especially in regard to display, control, and problem-solving.

The hardware configuration which provides the basis for the experience offered here includes an HP 3000 Precision Architecture (HPPA) computer (900 series) with a line printer (LP) and a laser printer (PP) in the computer room and other smaller line printers and laser printers located remotely.

Also, in this environment each printer is alone in its own device class. In other words, there is one printer in each device class so that Ldev and device class are (logically) equivalent for printers.

A printer can be directly connected to a computer with no mediation. In HP terminology, this is referred to as a "hot" printer. This is an inefficient situation even in the best of cases. One disadvantage is that the printer cannot be shared--it can only be accessed by one user at a time. Another disadvantage is that if the program is not executing, or if it is executing but not generating print, then the printer is idle when it potentially could be doing other work.

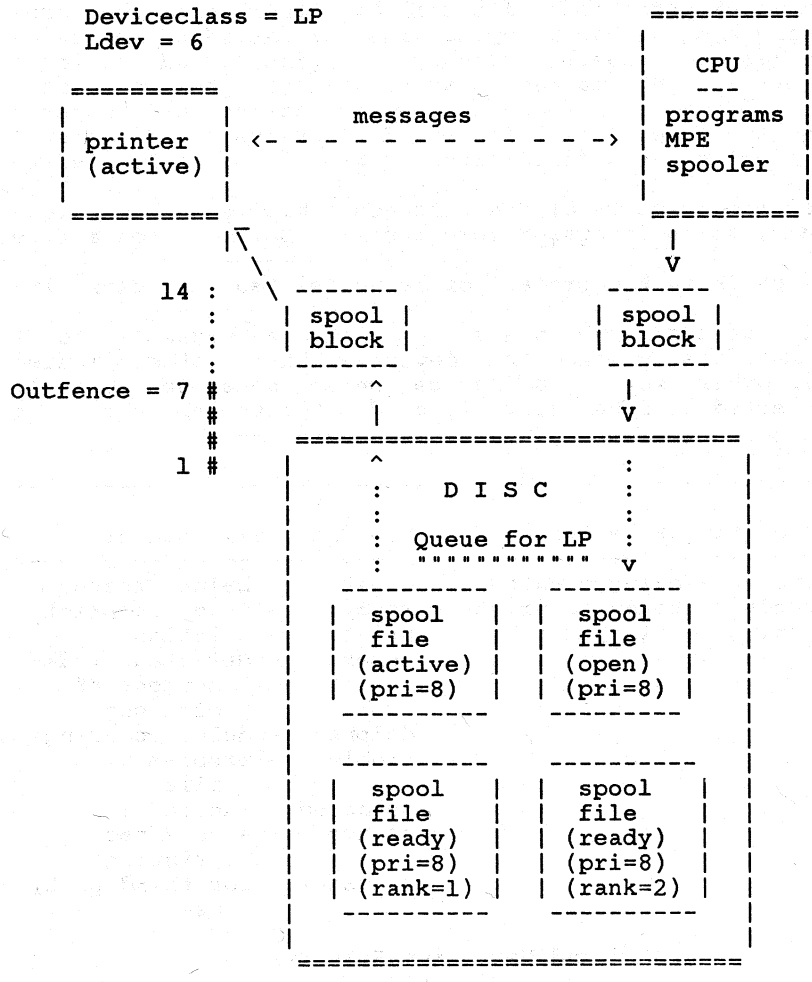
Spooling developed in order to allow more than

one user to access the printer at a time, to prevent unnecessary printer idle time, and to adjust to the increasing speed of the computer processor (CPU). Printing has increased in speed, but not nearly as fast as processor speed has. Since the electronics of the computer chips are now extremely fast compared to the electro-mechanical processes of the printer, the computer can generate output at a much faster rate than a printer (or even several printers) can print. In order to accommodate the printer, the computer holds the print files internally and feeds them to the printer as as it is ready for them. This is the spooling function. In practical terms, this means that there is a program or system process which stores print output as disc files and communicates with the printer about its progress and status.

There are certain mechanisms used in spooling. The files are kept in a waiting line (a queue) so that the first one created is the first one printed. An entire file is stored on disc and transferred to the printer in small blocks (in the HP case this is about two printed pages for a line printer, and more for a laser printer).

There are also specific constructs peculiar to the HP environment that are used to help the spooler do its work and to help the operator relate to the spooler (see Figure 1). The files are kept in queues, or waiting lines, for logical devices (Ldevs) and for device classes. A device class can contain more than one logical device (in other words, it is a set of devices), but it may have only one device. In this particular case, the spooler has a queue for the device class LP, which contains only one Ldev. LP is the HP default output device for jobs. (The default output device for sessions is a terminal.) Since there is only one printer in the device class, we can refer to this printer as the LP printer. The printer also has a logical device number of 6 (which is customary for the main printer on a system). Each print file has a status, or state, and an output priority number (or outpriority). There is an outfence, ranging in value from 1 to 14, which can be set by the operator. The status of the print file is (mostly) determined by how the

output priority compares to the outfence: files with priorities greater than the outfence are printed and files with priorities equal to or less than the outfence are deferred. Files which are waiting to print are ranked in the order in which they are to be print-



Spooling in the HP Environment
(Figure 1)

ed, according to their output priority.

In Native Mode spooling, there are more states for a print file than there were previously (see Figure 2). Formerly, a print file was "open" when it was being written to, "ready" when it was complete, "active" when it was being printed, "ready,d" when it was ready but deferred, and "locked" when a program (usually Spook, the spool file utility) other than the creating program had it open. The corresponding NM categories are "create," "ready," "print," and "defer." (There is no locked status--xfer is the closest.) In addition there are new categories of "defer," "delpnd," "problem," "selected," "spsave," and "xfer." "Defer" marks a file which has been explicitly deferred irrespective of the outfence (this is a new feature, to be discussed more later). Delpnd notes a file which is in the process of being deleted. Problem indicates a file which has a problem; an example is a print file assigned to a device which is not configured (in other words, a device which does not exist). Selected is apparently not used. Spsave labels a file

Old Print File States	New Print File States
open - being created	create - being created
ready - ready to print	ready - ready to print
active - printing	print - printing
	defer - deferred (independent of outfence)
	delpnd - delete pending
	problem - problem with file
	selected - (unused?)
	spsave - save after printing
	transfer - for third party use

Old and New Print File States
(Figure 2)

which is to be saved (rather than deleted) after printing. Xfer denotes a file which is being transferred between nodes on a network (this status is not used by HP--it is for third party software).

Display commands allow us to see what files the computer has (see Figure 3). The only listing command available formerly for print files was the Showout command. This command allows information to be selected according to several parameters, which helps present a more compact display.

A simple Showout command would select all spooled files from jobs, since one is usually more interested in job output. (The files excluded would mostly be session $STDLIST (output) files.) On a system with many deferred print files this display can easily exceed one screen. The volume of output makes the display more difficult to view and usually means there is more data being displayed than you want to see. A solution to this overabundance is to only display the data in which you are actually interested. For example, if you only wanted to see the ready non-deferred files, then you would use the "ready,n" parameter on the command.

The only printer display command formerly avail-

	MPE/V	MPE/XL, MPE/iX
List of Files:	Showout	Showout Spoolf Listspf
Display of File Content:	Spook5	Print Printspf any editor

File Display Commands
(Figure 3)

able was again the Showout command. The "dev" parameter allows only files for a particular device (say, a laser printer (PP)) to be displayed.

New commands available to display a list files are Spoolf and Listspf. Spoolf presents data in a manner similar to that of Showout, except that instead of showing Form, Space, and Rank, it shows Creator. Listspf adds to the Spoolf display by showing a summary which is similar to the summary on the Showout command. The Detail parameter of the Listspf command inserts a second line which has Jobname, space information, and create Date and Time.

Both the Spoof and the Listspf command have many parameters, thus making a large variety of command variations available. There are too many possibilities to cover here, but the selection equation in particular is very powerful. Any of the fields on the display may specified as selection criteria. Logical operators are also allowed; so, for example, one could have the construction:

```
SPOOLF O@;[SELEC= STATE=READY AND DEV=LP AND  &
   NOT(PRI=1)];SHOW
```

which will show all the ready files on the line printer that have an output priority other than one. User Defined Commands (UDC's) or command files can be of great benefit in this type of construction by eliminating some of the typing. For example, the previous sample could be abbreviated to:

```
SFSW "DEV=LP AND STATE=READY AND NOT(PRI=1)"
```

or to

```
SFSW "DE=LP + ST=R + N(PR=1)"
```

depending on how complicated you want your command file to be.

In brief, Spoolf is good for a concise display and Listspf is good if you want job name, file size, or date information--but in a longer display.

It should be noted that there are practical differences in the way the old and new commands are implemented. For Showout, you see all of the files on the system which meet the selection criteria, for example, all the active files. For the Spoolf and Listspf com-

mands, you see all the files for your logon ID which meet the selection criteria, for example, all your active files. The only way to see more is to have greater capability--account manager (AM), system supervisor (OP), or system manager (SM).

Another difference is that under the old spooling if you specify device class on a Showout command, it will show both files assigned to the device class and files assigned to the logical device(s) within the device class. However, under Native Mode Spooling, any command (whether old or new) will display exactly what is specified: if a device class is named, only files assigned to the device class will be shown (and not files on device(s) within the class); if logical device is supplied, then only files assigned to that device will be shown (and not files on the class of which the device is a part). In other words, now if you want to be sure to see everything assigned to a printer (both for the Ldev and the device class), then you have to use two commands--one for the Ldev and one for the device class.

Commands which display files allow us to see the content of a file. For the old spooling there is really only one method: Spook. For the new spooling there are at least four ways.

Spook is a spool file utility for the old spooling. It allows a user to display spool files, including carriage control information. Its mode of operation resembles HP's line editor--that is, it operates in line mode and is oriented to using line numbers.

In Native Mode Spooling, there are two commands which will display spool files to the screen: Printspf and Print. The Printspf command display a print file with carriage control information. Print is an existing MPE XL command which can now be used on spool files. It can be used to display a print file to a terminal screen, or to another location if it is redirected.

Printspf has the disadvantages that the data line is truncated on the screen, and that the carriage control displayed on the left side of the screen causes the data line to be even shorter. Print has a more more readable display because it does not show carriage

control and because it offers the data in screen-sized chunks (twenty-four lines at a time). Line ranges may be used to limit the output of both commands.

With the advent of NM Spooling, there are some programs which can now display print files. Any editor which can support variable length records can be used to display spool files. If the spool file is edited, then a copy has to be made--original spool files cannot be changed. The usefulness of any given editor for viewing and editing will depend upon the quality of the editor.

The Fcopy utility can also be used to display a print file to the screen by simply leaving the destination field blank, as in: FCOPY FROM=myfile;TO=.

There are a few commands which show the state of the printer. The older command is Showdev. This command used with a printer Ldev shows whether the device has spooling turned on or not by whether the entry "SPOOLER OUT" appears or not. This is of limited usefulness because it gives no indication of whether the device is suspended or not. This must be determined by the process of eliminating other possibilities. The newer command is Spooler. The Show parameter on the command allows the status of a printer to be displayed. This is a more informative command because it indicates different states of the printer, including idle, print, suspend, and suspend pending.

There are commands and programs available for manipulating the spool files. Some of these commands are old ones which still work, and two are new commands (see Figure 4). The old ones for print files are Altspoolfile and Deletespoolfile. These are both console commands which can only be used at the console terminal. Both are still usable, and Altspoolfile in particular is versatile because of its multiple parameters.

Spook is a spool file utility for the old spooling. It allows a user to display, print, search, delete, and combine spool files. Its mode of operation resembles HP's line editor. It is rather functional considering that it does not allow editing (only deletions, and only indirectly by making a new file). It does permit appending, copying, and deleting (both lines and whole files). It also has an option to display carriage control information.

	Old	Continuing	New
	===	==========	===
Commands:	-	Altspoolfile	Copy
		Deletespoolfile	Spoolf
Programs:	Spook	-	editors
			Fcopy
			Spfxfer

Old and New Print File Manipulation Tools
(Figure 4)

Spook has been eliminated for Native Mode Spooling, and its functions spread over several commands and programs. Even though there are now more options for displaying, some of the ease of use is lost because the functionality is fragmented rather than being integrated in one program as it was in Spook.

There is a new command for spool files. There are also some old tools with new abilities which enable you to manipulate spool files. The new command is Spoolf. We have already looked at Spoolf with regard to displaying spool files. However, in addition the Alter parameter allows you to change attributes of spool files. These include printer, output priority, copies, and status. In fact, the command is capable of mass changes, so it is possible, for example, to select all the files for the user Operator which have an output priority of 1 and are assigned to the page printer, and to alter them all to print immediately (assuming defaults) on the line printer with the command:

```
SPOOLF O@;SELEQ=[OWNER=OPERATOR.SYS AND  &
   DEV=PP AND PRI=1];ALTER;DEV=LP;PRI=8.
```

Here again, Spoolf is very flexible because of its many options.

With Native Mode Spooling, new abilities come into play with the Copy command, and with the Fcopy and editing programs. Copy is a command new for MPE/XL

which copies files from disc to disc. It now has the ability to copy spool files. Likewise, the Fcopy utility is able to copy spool files in any manner which it can copy other files. Any text editing program (including Edit/3000) which can handle variable record length files is now able to edit spool files (allowing for existing file size limitations). We will see the reason for these new abilities later when we look at the change in the nature of spool files.

There is a new utility program called Spfxfer which comes with NM Spooling. It can read spool files from a Spook Output tape (Compatability Mode) and restore them on an MPE/XL or MPE/iX system with NMS as Native Mode spool files. Likewise, it can put Native Mode spool files from an MPE/XL or MPE/iX system onto tape in the Spook Output format. This allows spool files to be transferred both ways between systems which have the two different modes of spooling.

Having these new spool file commands and utilities available gives a much broader ability to manipulate the contents and location of spool files.

Continuing	New
==========	===
Resumespool	Spooler
Startspool	
Stopspool	
Suspendspool	

Printer Control Commands
(Figure 5)

From the standpoint of controlling the printer, there are again both old and new commands which are available (see Figure 5). The old commands are: Resumespool, Startspool, Stopspool, and Suspendspool. These still perform their original functions of resuming, starting, stopping, and suspending the spooling operation. Startspool allows the printer to print from the spool queue, Stopspool halts printing (while leaving the queue open to accumulate files), Suspendspool pauses the printer, and Resumespool begins printing after a suspend. These are all console commands which must be used at the console terminal.

All of these spool functions (and more) have been incorporated into a single new command, Spooler. Spooler has the ability to: open and shut spool queues (which we will not discuss further on the assumption that the queue is always left open); start, stop, suspend, and resume spooling; and show printer status. Spooler also has the ability to release a kept file and to reset the file position pointer, both of which will be covered more when we view changes to spooling and the application of those changes. Here again, as with Spoolf, the number of options on the command lends it to being incorporated into UDC's or command files. In dealing with the printer, Spooler, along with Spoolf, can be used to handle most circumstances.

We will now consider some of the features or differences of Native Mode Spooling over Compatability Mode Spooling. These include: permanence of spool files, keep option on suspend, storable pointer, immediate suspension, relative or absolute offset, independent deferral, printer status display, altered header format, and file copying (see Figure 6).

A major change is that spool files are now permanent files. Formerly, output spool files were entered

---

Permanence of Spool Files

Keep Option on Suspend

Storable Pointer

Immediate Suspend

Relative or Absolute Offset

Independent Defferal

Printer Status Display

Altered Header

File Copying

Features/Differences of Native Mode Spooling
(Figure 6)

---

into their own directory (the OOD), which was maintained separately from the System Directory. Now, spool files are recorded in the System Directory. They are stored in the Account HPSPOOL; input spool files are in the IN group, and output spool files are in the OUT group. The file names consist of an "I" and a number for input files (such as I7251), or an "O" and a number for output files (such as O342). So the complete name for an output file would look like: O342.OUT.HPSPOOL. Output spool files have a file type of OUTSP. UDC's or command files are helpful in accessing spool files because the digit portion is the only part of the file name which changes. Additionally information on input/output devices in kept in an account called 3000DEVS. Spool files as permanent files in the System Directory is an important change which allows editing, purging, copying, storing, and all other normal file operations. We will examine these operations separately below.

Another large change is the choice of keeping or releasing a file on a suspend, which is offered as an option on the Spooler command. Release (or Nokeep) puts the file at the end of the printer queue as formerly always occurred with suspend. Keep, which is the default, causes a file to be held at the top of the queue for the printer, instead of being dropped to the bottom. This approach makes it easier to suspend and resume a printer. Additionally, the Spooler command provides the option, between a suspend (with keep) and a resume, of releasing the kept file. This means that the file is put at the bottom of the of the printer queue, as formerly happened as part of a suspend.

One very useful new feature is the increased availability of the spool file pointer. As the computer prints a file, it keeps a pointer to tell it how far along the printing has progressed. Previously, this pointer was only used for the Back and Forward options on the Resumespool command. If the file printing was suspended and the file deferred, then the pointer was reset and any later printing began at the beginning of the file again. Now this pointer is available to the Offset parameter of the Spooler command, but in addition, if the file printing is suspended and the file is deferred, then the pointer is stored in the file label extension and can be used to begin printing at the place where printing was terminated, rather than at the beginning, when the file is printed again. This can save much unnecessary printing duplication.

Another difference is in the timing of a suspend. For Compatibility Mode, a Suspendspool command terminates the printing at the end of a page, prints an Incomplete trailer, puts the printer in Suspend status, puts the file in Ready status, and places the file at the end of the printer queue for its output priority. For Native Mode, on a Spooler command with the Suspend option the printing stops wherever it is on the page (for a line printer), there is no trailer page printed (unless the file is released), the printer is put in Suspend state, the file remains in Print state, and the file remains at the top of the printer queue. This approach allows printing to begin after a Spooler command with the Resume option as if there had been no interruption.

The Back and Forward options on the Resumespool command are replaced by the Offset option on the Spooler command. Previously, the Back and Forward were relative to the position at the suspend, but now with Offset the positioning is relative or absolute. If a plus sign (+) or a minus sign (-) is used with the number, then the positioning is that number of pages forward or backward from the place of the suspend (which is relative to the suspend location). If there is no plus or minus sign, but simply a number, then the positioning is that number of pages from the beginning of the file (which is absolute with respect to the beginning of the file. Beware: if you forward past the end of the file, the system assumes that you are done with the file and deletes it.

There is also now a difference in the way a print file can be deferred. In the past, the status of a print file was determined by comparing its output priority to the outfence setting of the printer to which it was assigned. With NM Spooling this is still true, but a spool file can also be deferred independent of the outfence by simply changing its status to Defer. This is accomplished by using the Defer and Undefer options on the Spoolf command. It is now possible to have an outfence of 7 and a spool file with an output priority of 8, yet have the spool file be deferred because it was assigned that state independent of the outfence setting.

The printer status is now more accessible. The only condition which could be determined formerly was whether a printer had spooling started or stopped. This information was available through the Showdev command which showed a "SPOOLER OUT" field for a printer

which had spooling started, and a blank field for a printer which had spooling stopped. This was the extent of displaying the status of a printer. The new Spooler command has a Show option which explicitly lists the status of a printer. The SPSTATE field will show the states Idle, Print, Suspend, and Suspend Pending. This is particularly nice because it was not possible before to directly determine whether a printer was suspended. The Print state corresponds to the former Active state.

The header page format has been altered. The header page for copies of a spool file were all the same, except perhaps for the time printed. In NM Spooling there is an extra field at the end of the header line to show the copy number. For a single-copy file, the field shows "1 of 1". For a file with, for example, three copies, the field on the header of the first copy would show "1 of 3", the field on the header of the second copy would show "2 of 3", and the header of the third would show "3 of 3".

Formerly, spool files were part of the spooling system and could not be copied into the file system. The Spook utility program could be used to make copies of spool files, but the copies were still part of the spooling system rather than the file system. Now that spool files are in the file system (by being in the System Directory), it is possible to copy spool files into other parts of the file system (that is, into other accounts). However, the spool system needs a means of determining which files are the original spool files and which files are the copies. To do this, it keeps a list of the original spool files. HP refers to these files as being "linked" to the spooling system. In turn, spool files copies not in the HPSPOOL account are refered to as unlinked.

From an operational standpoint, the Native Mode Spooling changes that have the biggest impact are: immediate suspension with the file kept active; the keep/release file option on suspend; storage of the pointer on suspend and deferal of the file; and the ability to defer a spool file independently of the outfence.

In terms of applying the new features to spool files and printers, one of the most common situations is a paper jam. On jams which the printer detects immediately, recovery is a matter of realigning the form and reseting top of form. When the printer is put online, it will automatically reprint the page. In a

case where the printer does not detect a printing fault or does not detect it immediately, then more than one page will need to be reprinted. In this case, the condition needs to be corrected (that is the form and printer realigned), the printer suspended, and then the printer resumed using the Back option with the negative number of pages to be reprinted (plus a few more for a margin of error).

If a printer is not functioning, the first thing check is whether it is on-line or not. If it is off-line, it may simply need to be put on-line. If it is already on-line, then it may be stopped or suspended. Use the Spooler command to display the status. If it is stopped, then it needs to be restarted (with Spooler and Start); if it is suspended, then it needs to be resumed (with Spooler and Resume). If a printer has timed out and stopped, then the condition which caused the time-out (off-line, out of paper, paper jam, etc.) needs to be corrected before it is restarted or it will simply time out again.

If you have a case where a large file is printing, and you urgently need to print another file on the same printer, instead of waiting for the first file to end or interrupting the first one and having to start it from the beginning later, it is possible with NM Spooling to suspend printing, release the first file (which causes the pointer to be saved), defer the first file for later, alter the second file to the top of the queue if it is not there already, resume the printer, and print the second file. The first file can then be printed later beginning from the place where it was suspended simply by altering its priority to enable it to print.

It is possible to put escape code instructions for a printer inside a print file. These are often simply one line at the beginning of the file to set the printer for the coming file. At the end of the printing, the printer reverts to the defaults. It can happen that these settings would be lost, say during a reset or a power failure. The old approach would be to reprint the file from the beginning, or at least to restart it from the beginning and then to resume forward, in order to set the printer with the special settings for the print file. With NM Spooling, it is possible to edit the file to leave the escape codes at the beginning, and to delete the portion of the file which has already printed, so that printing can proceed from

the place where it was interrupted.

There are several important consequences of the fact that spool files are permanent files. They may be stored, edited, and copied, and of course, they are permanent. Since spool files are in the System Directory, they may be stored and restored. This includes both as a separate operation and as part of a system backup. One benefit of this situation is that if a spool file is stored and the computer copy is accidentally deleted or needs to be reprinted, then the spool file can be restored from tape and reprinted. A disadvantage is that a wholesale restore of spool files from a store tape will result in the reappearance of spool files which printed normally and were purged by the spooling system.

Spool files under Native Mode Spooling may also be edited. Since they are in the System Directory, they may be referenced by any editor. However, you are in effect editing a copy because you cannot save back to the original file. This is merely an inconvenience because the copy can be directed to a printer and become a spool file in its own right. One thing to watch during editing or copying is that any carriage control is not lost. If the CCTL parameter or the OUTSP file type are not used, then the carriage control characters loose their significance and become part of the data to be printed.

Spool files may be copied under NM Spooling. The complication here is that copies, whether edited or unedited, will not be printed unless they are directed back to a printer. HP's way of saying this is that spool files are "linked" to the spooling system, and that copies are "unlinked" (that is, will not automatically print). The way around this, of course, is that if you want the copy to print that you send it to a printer. Here again, care must be taken with spool file copies to preserve their carriage control.

Spool files under NM Spooling are permanent. They will be deleted after printing as before (except for Spsave files), but they will no longer be lost on some system startups. This is because spool files were previously tracked by having their own directory, which was rebuilt after every startup but a Start Norecovery (that is, a Warmstart). Now, spool files are kept in the System Directory. This means that they are not removed unless they are explcitly deleted.

One technique to use on NM spool files is to renumber them by storing them separately with the purge option, then restoring them. This will make all the create dates on a Listspf the same (although the original date is preserved in Listf in the "3" format), but it will force a renumbering starting with 1 again of all the spool files. This looses the continuity of the filename, but is a way of compacting the spool file numbering and thus keeping the numbers lower than they would otherwise be. This is an advantage because smaller numbers are easier to use.

There are some things which are unchanged under NM Spooling. All the old printer related commands still work; however the Spook utility is obsolete.

Serial time-outs can still occur; parallel printers still do not time out.

Recovery status (status 2) on line printers is still an affair of unpredictable time length.

Association is still available between a printer and a user ID. Use of the Associate command, of an HP utility program to create or rebuild a system table, and of a log-on UDC will allow an individual user to control a printer in the same way in which the operator can. This can even be simplified for the user by employing menus to save typing and memorization.

System logging can be turned on to log spool files which actually print; non-printed files are not logged.
----- -------- -----
The HP Native Mode Utility Logtool is available to format records from system log files into a printed report.

Some obvious benefits of Native Mode Spooling are: Boolean selection on spool file display; explicit printer status display; editing, copying, and storing of spool files; permanence of spool files; storage of the spool pointer; and copy notation on the header page.

On the whole, Native Mode Spooling brings more flexibility and utility than its predecessor--at the expense of having to learn to do most things differently.

Paper #5035

Designing an INTRINSIC Procedure in Your Favorite Language

Craig Nickerson
United Electric Controls Co.
180 Dexter Ave.
P.O. Box 9143
Watertown, MA 02272-9143
U.S.A.
(617) 926-1000

Introduction

In the course of writing general-purpose library procedures for our Classic MPE shop, I have found it appropriate to make a number of them declarable as intrinsics.

Apart from making calls to these procedures easier to code, especially where value parameters are concerned, I have found another reason; I write a fair amount of my stuff in FORTRAN 77.

So what?

Well, originally the "IF(CCODE())..." construct worked exactly like "IF(.CC.)..." in FORTRAN 66. Then, at some point HP re-designed the internals, so that now, "IF(CCODE())..." only works for declared SYSTEM INTRINSICs. We have a workaround whereby the Condition Code from a non-intrinsic is saved and retrieved through the X register[1] but it doesn't work for declared intrinsics. I prefer to have a uniform way of doing things.

Because our shop does not have, and has never had, the SPL compiler (at which a number of my colleagues in the profession have expressed surprise), I despaired of intrinsicizing my procedures, until I stumbled on the solution in, of all places, VESOFT's SECURITY/3000[2] manual. I therein discovered that standard equipment on Classic systems is a software-development tool that "compiles" an SPL intrinsic definition into a SPLINTR file[3]

In this paper, I will show you how a declarable intrinsic could be written in FORTRAN 77. The basic techniques could be applied to whatever third-generation language you prefer to work with.

What Is an "Intrinsic"?

An *intrinsic* is any procedure that can be declared as such in 3GL source code using a language-supported construct, by virtue of having a definition header in an SPL INTRinsic (SPLINTR) file, accessible to the compiler and to which your logon has READ access. In this way, the compiler knows the type and number of parameters, the type of procedure, and how to generate the correct code to call it, making your source code that much easier to write.

Although all documented *system intrinsics* reside in the system library (SL.PUB.SYS), an intrinsic may reside in any SL, or even in a program segment; nor does residence in the system library in and of itself make a procedure available to INTRINSIC constructs.

Furthermore, although all system intrinsics were written in SPL, an intrinsic may be coded in any 3GL (just about), subject to whatever limitations are inherent in the language.

The formal definition of a procedure as an intrinsic, although required to be coded in SPL, can be a completely separate and independently-produced entity from the object code.

Using FORTRAN 77: Some General Considerations

As I said in the "Introduction", our frame of reference will be FORTRAN 77.

The parameters passed to intrinsics must structurally fit the data types supported by SPL, so your directives should include the following:

$SHORT
Type INTEGER data in intrinsics is assumed to be single-word (INTEGER*2); type INTEGER*4 (the default integer size) corresponds to SPL type DOUBLE.

$FTN3000_66 CHARS [ON]
INTRINSIC constructs do not provide for string descriptors as such; byte address and length by value must be passed as separate parameters.

$FTN3000_66 LOGICALS
This makes all type LOGICAL data and functions compatible with SPL type LOGICAL. (I haven't seen anything on this in any of the FORTRAN 77 manuals; I got it from *HP3000 Application Note #84.*)

$CHECK_FORMAL_PARM n
If all of your parameters are by reference, your parameter and procedure types compatible with all anticipated callers, and your procedure is not "OPTION VARIABLE", you may use the

default level 3. A level ≤ 2 is recommended if any parameters are passed by value, or your procedure is "OPTION VARIABLE". If any value parameters are longer than one word, such as type INTEGER*4 (DOUBLE) or REAL, you must use a level ≤ 1 to suppress checking of the number of parameters. Level 0 is mandatory if, for any reason, your formal procedure type must differ from the defined actual[4]

If object file directory space is a major concern, you should, on general principles, compile all of your intrinsics under "$CHECK_FORMAL_PARM 0" (which is, in fact, the default checking level in SPL).

The underscore (_) and dollar sign ($) characters legal in FORTRAN 77 symbols are not legal in SPL symbols.

Since SPL supports only non-ARRAY typed procedures, an intrinsic function cannot be defined as equivalent to a type LOGICAL*4 or any type CHARACTER or COMPLEX.

Alternate Return label parameters are unique to FORTRAN and cannot be used in intrinsics.

Value Parameters

In any 3GL, the constructs for interfacing with intrinsics support the passage of any non-array parameter *by value* as opposed to *by reference*. When a parameter is passed by value, the parameter list stacked by compiler-generated code contains its value instead of its DB-relative stack address.

This presents a technical problem for FORTRAN: All formal parameters (traditionally called "dummy arguments") are, in effect, required to have been passed by reference; language rules support value parameters only where they are actual, i.e. in a CALL statement or implied call to a function.

To overcome this obstacle, FORTRAN 77 provides two options that empower you to get at a passed value parameter so that you can work with it locally. One is to type it as a CHARACTER argument and pass it to the pseudo-function BADDRESS, which is fooled into thinking it's a byte pointer. The more universal solution, which may be shared by intrinsics coded in other 3GLs, is the following procedure:

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT,FTN3000_66 CHARS
$CHECK_FORMAL_PARM 2
C
      INTEGER FUNCTION GETVAL(VP)
C RETRIEVES BY-VALUE FORMAL PARAMETER.
C
      CHARACTER VP*2
C
      GETVAL=BADDRESS(VP)
```

```
      RETURN
      END
```

Here's how it's used:

```
      SUBROUTINE TESTINT1(parm1,parm2,parm3,VARG,
     *parm5)
      ...
      INTEGER VARG,GETVAL,IARG
      ...
      IARG=GETVAL(VARG)
```

Dummy argument VARG stands in lieu of a value parameter of type INTEGER. The object code, of course, thinks that its berth in the parameter list is occupied by a stack address, so any attempt to reference VARG directly in any sort of arithmetic expression is liable to result in a "BOUNDS VIOLATION" trap, or some other unpredictable error. BUT--it also thinks that it's being passed by reference to GETVAL; so, the object code simply copies what it thinks is an address into the parameter list for GETVAL, without any inkling of what GETVAL is really up to! Thus, GETVAL winds up getting the datum by value masquerading as a byte pointer, and so it comes to rest in local variable IARG where we are now in a position to use it.

Other data types like INTEGER*4 (DOUBLE), REAL and REAL*8 (DOUBLE PRECISION or LONG) occupy two or more words each in memory; consequently, when passed by value, they occupy the same number of words in the parameter list. These must be covered in your procedure code by a separate INTEGER argument for each word of data, to be retrieved through a separate GETVAL call; for this reason, your formal checking level must be ≤ 1.

For example, a type REAL by value would be handled like this:

```
      SUBROUTINE TESTINT2(parm1,VRW1,VRW2,parm3,parm4)
      ...
      INTEGER VRW1,VRW2,GETVAL,IVR(2)
      REAL XR
      EQUIVALENCE (XR,IVR)
      ...
      IVR(1)=GETVAL(VRW1)
      IVR(2)=GETVAL(VRW2)
```

This intrinsic has four (4) parameters by definition, of which the second, covered by VRW1 and VRW2, is REAL by value. Local REAL variable XR is loaded word-by-word using overlaid INTEGER array IVR. The other multi-word data types can be handled in similar fashion.

Programming for "OPTION VARIABLE"

The Intrinsics manual contains many examples of intrinsics that are "OPTION VARIABLE"--meaning that some parameters are *required*,

and some are *optional*; the latter kind can be physically omitted from the calling sequence in source code.

To avail yourself of this feature, your procedure code must provide for an additional value parameter called a *parameter mask*; this is a bit map indicating the presence or absence of each data parameter (not counting itself), and always occurs in your SUBROUTINE or FUNCTION statement as the last argument. The mask bits are assigned from the right-most bit, applying to the last data parameter, and proceeding left. For each bit, a value of 1 means that the corresponding data parameter is "present", while 0 means "absent", or "omitted".

To accommodate more than 16 parameters, the mask must be covered by more than one argument; the exact size in words is given by

```
(<number of parameters> + 15) / 16
```

The last 16 parameters are covered by the last word of the mask, the method of bit assignment being uniform in all cases.

When you call a declared intrinsic that is "OPTION VARIABLE", you don't have to worry about the parameter mask, because the code for setting it up is compiled automatically. However, the compiler has no way of distinguishing between required and optional parameters; the presence of each must be checked, and enforced as necessary, at run time by the intrinsic itself, and you are responsible for the code needed to do this.

Bear in mind also that although two or more formal (INTEGER) arguments are needed to capture a multi-word datum passed by value, in a parameter mask, they count collectively as one parameter.

Since parameter masks are always passed by value, this is another area where our old friend GETVAL comes in very handy indeed.

Parameterless Functions

A function--or *typed procedure* in SPL terminology--is not absolutely required to have a parameter list. FATHER and GETJCW are two examples of MPE intrinsic functions that are "parameterless".

Nevertheless, FORTRAN 66 is one 3GL that requires all non-intrinsic functions to have at least one parameter. The language manual suggests a workaround whereby a call to a function with no parameters may be coded with any single argument, and encapsulated in a subroutine in order to clear the extra word left at the top-of-stack[5]--but this only works if the called function's formal checking level is less than 2.

Fortunately, the parameterless function is a legal construct in FORTRAN 77, so making an intrinsic of one, to make it cleanly

callable from FORTRAN 66, presents no special difficulty.

Designated Procedures

The internals of passing subroutine and function names (type EXTERNAL symbols) as actual arguments are very different from FORTRAN 66 and not compatible with SPL subtype PROCEDURE; this is why FORTRAN 77, as far as release A.01.00, has had a problem with "P"-type parameters in such MPE intrinsics as SORTINIT.

For compatibility with all 3GLs, a procedure-name parameter must be defined and handled as an integer *plabel* by value. If you intend for the designated procedure to be called from your intrinsic, then once GETVALed into a local variable, the *plabel* may be passed by reference to a FORTRAN 77 "gateway" procedure6.

Returning a Condition Code

No discourse on intrinsics is complete without a discussion of the *Condition Code*--the means whereby an intrinsic gives its caller a rough idea as to the success or failure of its mission, without using a passed parameter.

The Condition Code is a 2-bit field in the status register which helps to indicate the outcome of the most recently executed instruction. Not all instructions affect the Condition Code, but it can be meaningful for PCAL, for a called procedure may return a Condition Code by setting the corresponding field in the old status word at Q-1 in its "above" stack marker at any time before it EXITs.

A Condition Code value is expressed in terms of the comparison of some imaginary value to 0, and the assignment of these terms to the corresponding field values is as follows:

0 CCG (greater than); usually means that a routine exception occurred, such as end-of-file.

1 CCL (less than); usually indicates the occurrence of an unexpected error.

2 CCE (equal to); usually means "mission accomplished".

Since SPL is the only language that supports the necessary operations for manipulating a stack marker, your FORTRAN 77 intrinsic will need a support procedure.

A procedure call, of course, generates another stack marker, so your SETCCODE procedure must trace back to your intrinsic's above stack marker by means of the delta-Q saved in Q-0; in other words, it must set the desired Condition Code in bits (6:2) at location Q-'Q-0'-1.

Eugene Volokh gives an example of a Condition Code support procedure in his paper "Making Other People's Programs Do What They Were Never Intended to Do"[7].

Preparing the Intrinsic Definition

Formally defining your procedure as an intrinsic is surprisingly simple--all you have to do is write an SPL header for your procedure and compile it into a SPLINTR file using the system program BUILDINT.PUB.SYS.

Writing the Header

An intrinsic header has the following general form:

```
[proc-type ]PROCEDURE proc-name[ (parmlist-1)];
[VALUE parmlist-2;]
[data-type-1[ ARRAY] parmlist-3;]
[data-type-2[ ARRAY] parmlist-4;]
...
[data-type-M[ ARRAY] parmlist-N;]
OPTION EXTERNAL[, VARIABLE][, CHECK level];
```

Proc-type must be present if the intrinsic is a function, and omitted if it is a subroutine. *Proc-type* may be one of the following:

INTEGER
LOGICAL
DOUBLE
REAL
LONG

Parmlist-1 through *parmlist-N* each consists of one or more simple variable or array names separated by a comma (,) and at least one space. These names need not match the names used in your code. *Parmlist-1* does not include a parameter mask.

Parmlist-2 is a subset of *parmlist-1* indicating which simple (non-array) parameters are passed by value; all others are presumed to be by reference. The VALUE clause must appear before any data-typing clause. Value parameters of any type must be represented by single arguments appropriately typed.

Parmlist-3 and *parmlist-4* are mutually exclusive subsets of *parmlist-1* indicating which parameters are of *data-type-1* or *data-type-2*, respectively. In addition to the keywords allowed for *proc-type*, you may also specify "BYTE ARRAY"; the keyword "ARRAY" may be applied to any other data type, as appropriate.

Level should be the same as the formal checking level used in your code; the default is 0.

Compiling the Header

The FORTRAN 77, SPL and PASCAL compilers can work from any SPLINTR file of your choice, but if you want your intrinsic to be globally declarable in all languages, including FORTRAN 66 and COBOL II, you must set it up in SPLINTR.PUB.SYS.

Program BUILDINT.PUB.SYS reads from $STDIN one or more intrinsic headers separated by a blank or null line, the entire set terminated by "END.". After checking them for proper SPL syntax, it compiles them into the SPLINTR file, overwriting any pre-existing definitions of the same procedure name. It then produces a report on $STDLIST of all intrinsics currently defined in the SPLINTR file.

To install your new system intrinsic definition(s), you must logon with System Manager (SM) capability, and equate formal designator SPLINTR to the system SPLINTR file before you run BUILDINT. You can enter the header from the terminal if it's simple enough, but the least little typographical error can cause a lot of your valuable time to be wasted; better to write it into an unnumbered file and use the :RUN command's ;STDIN= option. In any case, $STDLIST should be diverted to a printer device file with carriage-control.

WARNING: BUILDINT overwrites any entries in the file with duplicate intrinsic names without checking with you first! So, you had better make sure that your intrinsic names are not the same as MPE's or anyone else's. If you enter only "END." and nothing else, BUILDINT will leave the SPLINTR file unchanged and just give you an alphabetical list of what's currently out there.

It would also be wise to save all of your intrinsic headers in a :JOB stream that would be ready to run right after an operating system upgrade (can you guess why?).

Example:

GENITEM is a FORTRAN 77 application procedure that I recently added to our third-party manufacturing software. Used in certain product/component inquiry functions, it encapsulates the chores of prompting the user for either an exact part number or a part set (using the familiar MPE "wild cards"), and gathering all part numbers that match a given set. Since I wanted some passed parameters to be optional, because they may or may not be useful to the calling application, and also wanted to return a Condition Code to facilitate error checking, I have designed and installed it as a declarable intrinsic.

Here is the calling sequence as it might have been described in the Intrinsics manual:

```
  O-V      IA      I      BA       LV       L       L
GENITEM(partno,error,prefix,allowall,setflag,oldset,

          BA
        genpn);
```

partno	*integer array (required)*
error	*integer (required)*
prefix	*byte array (optional)*
allowall	*logical by value (optional)*
setflag	*logical (optional)*
oldset	*logical (optional)*
genpn	*byte array (optional)*

The parameter mask ANDed with %140 yields a result of %140 if and only if both required parameters are present.

This is the intrinsic definition header that I compiled into the system SPLINTR file:

```
PROCEDURE GENITEM (P1, P2, P3, P4, P5, P6, P7);
VALUE P4;
INTEGER ARRAY P1;
INTEGER P2;
BYTE ARRAY P3, P7;
LOGICAL P4, P5, P6;
OPTION EXTERNAL, VARIABLE, CHECK 2;
END.
```

Note that the parameter mask is not included among the formal parameters; it is provided for by the keyword "VARIABLE" in the OPTION list.

A Brief Look at Other 3GLs

COBOL II

Formal and actual parameter checking are both fixed at 2 and cannot be altered; formal parameters are limited to non-character types; and the program unit as typed procedure is not even a legal construct. But single-word value parameters and "OPTION VARIABLE" are made feasible by the pseudo-intrinsic .LOC.. Intrinsicizing is desirable, in any case, if you're returning a Condition Code and want full compatibility with FORTRAN 77 callers.

FORTRAN 66

Much less restrictive than COBOL II, this compiler lets you do almost everything possible in FORTRAN 77, except that reading value parameters is a bit more problematical. HP has documented a way to use the ASCII intrinsic to get byte pointers for CREATEPROCESS calls, which may be applied to the present problem, but it would collide with conventional ASCII calls in the same code segment. Anyway, it would be much more efficient to enlist the aid of another compiler; you could have as many as four to choose from.

Also, the parameterless function requires some trickery--not to mention $CONTROL option CHECK=0.

Just on the other side of the parameter list from a function's above stack marker and its local data, the caller's object code has allocated sufficient space on the stack for the returned value, the number of words being appropriate to the function type. After RETURN is executed, the caller "pops" this value off the stack before going on to the next statement.

The trick, then, is to code your parameterless function as a parameterless SUBROUTINE, using SPL support procedures to find the Q (DB-relative address) of your above stack marker, and to store data to a calculated stack address, thus emulating the functionality of a bona fide typed procedure.

For example, to pinch-hit for a parameterless function of type INTEGER*4 (DOUBLE), a subroutine must return the high-order and low-order words of the function value to locations Q-5 and Q-4, respectively.

Conclusion

The benefits of INTRINSIC constructs are not the exclusive property of the documented intrinsics, nor even of SPL procedures generally. With little additional effort, and minimal risk to the integrity of your system, your own non-SPL programs can "share the wealth".

I should add that the foregoing techniques may be applied to the strategy of hooking system intrinsics. However, in the case of such time-critical procedures as FREAD that have value parameters, you had best use SPL, if possible.

* * * * *

This paper is dedicated to

KEN ORTON

NOTES

1. See my paper "Secrets of MPE V Tables III: Square Pegs for Round Holes"; 1993 INTEREX Proceedings, San Francisco, CA; Paper #5042.

2. For more details on VESOFT products, contact:

   VESOFT, Inc.
   1135 S. Beverly Dr.
   Los Angeles, CA 90035-1119

3. Page 207 of the SECURITY/3000 manual mentions a job stream that runs the program to add intrinsic definitions for the related procedures.

4. My INTEREX '91 paper on FORTRAN's Alternate Return constructs contains an unfortunate misstatement regarding "procedure" vs. "function" type. The truth is, different function types are different procedure types. When writing an INTEREX paper, you cannot research your subject too carefully!

5. FORTRAN/3000 Reference Manual, Section A-2.

6. I describe this technique in Appendix I of "MORE Secrets of MPE V Tables: A Closer Look at Code Management Structures"; 1992 INTEREX Proceedings, New Orleans, LA; Paper #3051.

7. 1990 INTEREX Proceedings, Boston, MA; Paper #3141.

   I think Eugene's procedure succeeds in User Mode because array elements are addressed via the X register. My earliest attempt at a support procedure for Condition Codes used indirect addressing only and got socked with a "BOUNDS VIOLATION".

Paper # 5037

What's New with MPE V?

Robert Holdsworth

Hewlett-Packard
Software Technology Division
Roseville, California
916-786-8000

Introduction

This paper discusses enhancements made to the MPE V Fundamental Operating System (FOS) on Releases 30 and 31, by Hewlett-Packard Software Technology Division (SWT). SWT identifies customer priorities by closely monitoring service requests, through involvement with user groups and the System Improvement Committee, and by contacting customers directly. You can look forward to more enhancements with future releases of MPE V, and you are encouraged to actively voice your recommendations for new development.

Send ideas for future enhahcements to MPE V to:

Bob Stamps
Hewlett-Packard Company
Software Technology Division
8000 Foothills Blvd., MS R5YB
Roseville, CA, 95747-9987
USA

MPE V Release 30

Purge Command Accepts Wildcard

One of the most popular enhancements to Release 30 (based on feedback from test sites) was the ability of the PURGE command to allow multiple files to be deleted. The PURGE command now accepts the standard wildcard characters, used by the LISTF command, in the specification of the files to be deleted. Several new options were added to the PURGE command to allow

flexibility in deleting multiple files. These options are shown in the new syntax of the command below and are discussed in more detail below.

```
PURGE fileset [{,}TEMP]  [{;AUTOLOCKWORD   }]
               {;}        {;NOAUTOLOCKWORD}]
              [{;CONFIRM   }][{;NOSHOW}][{;NOSHOWERROR}]
               {;NOCONFIRM } {;SHOW   }  {;SHOWERROR  }
               {;CONFIRMALL}
              [;ONERROR={CONTINUE}]
                        {QUIT    }
              [;ONLOCKWORD={SELECT}]
                           {SKIP  }
```

The AUTOLOCKWORD option permits SM/AM users to purge files without having to respond to lockword prompts. For AM users, AUTOLOCKWORD is effective only for the files in the user logon account. The default is NOAUTOLOCKWORD.

CONFIRM, NOCONFIRM, and COMFIRMALL allows the user to specify the desired level of confirmation for PURGE. CONFIRM results in one prompt to verify that the fileset specified is correct; this is the default in interactive mode. CONFIRMALL results in a verification prompt for each file within the fileset. NOCONFIRM results in no verification prompt; this is the default in batch mode.

The SHOW/NOSHOW option allows the user to control whether the filenames are displayed to $STDLIST as they are deleted. If the SHOW value is chosen for this option the output can be written to a file by redirecting the formal file designator SYSLIST. NOSHOW is the default.

The SHOWERROR/NOSHOWERROR option controls the detail of error messages that are displayed. NOSHOWERROR is the default.

The ONERROR keyword parameter was added to allow the user to control whether or not the PURGE command should continue upon encountering an error. The default is to CONTINUE executing the PURGE command upon encountering an error. The other option is to specify ONERROR=QUIT.

ONLOCKWORD allows the user to control whether or not files with lockwords are deleted or skipped. This option is available for all the users.

ALTSEC Command Accepts Wildcard for ACDs

The ALTSEC command has been enhanced to allow the manipulation of ACDs (access control definitions) for a set of files. This feature is offered on MPE/iX and now it is available on MPE V. ACD manipulation used with one of the following options: NEWACD, COPYACD, ADDPAIR, REPPAIR, DELPAIR, or DELACD accepts the standard wildcard characters, used by the LISTF command, in the specification of the files to be altered. Note that MPE's conventional file access security matrix manipulation is not affected by this enhancement. It still expects one file at a time.

The ACD warning status 106 from the HPACDPUT intrinsic and the CIWARN 7106 from the ALTSEC command have been reassigned to 107 and 7107 respectively in order to maintain the compatibility with MPE/iX. The ACD warning 107 and the CIWARN 7107 mean "PSEUDO EXTENT POINTER WAS CORRUPTED PRIOR TO BEING DELETED." This warning is generated when the corruption of the pointer to the "pseudo extent" is detected in deleting an ACD. A "pseudo extent" is where the system maintains ACD information for each file or device. Upon encountering this warning, the delete operation succeeds, so there is no longer an ACD associated with the file or device, and the pointer no longer contains an illegal value. Programs / jobs that delete ACDs using HPACDPUT / ALTSEC need to be modified if they make explicit checks for these warning values. With the introduction of wildcarding capability in ACD manipulation in the ALTSEC command, CIWARN 7106 now means "OPERATION FAILED ON SOME FILES" as on MPE/iX.

New LISTF Options (-3, 3, 4, and 6)

The LISTF command has been enhanced to provide the following optional information levels: -3, 3, 4, and 6. Options 3 and -3 are equivalent to the information provided by the "listf" command in the LISTDIR5 utility, as is the option 4 equivalent to the "listsec" command in the LISTDIR5 utility. The information presented with options 3, 4, 6, and -3 has been formatted to closely match the formatted output for the MPE/iX LISTF commands. The options provide additional file label information as well as security provisions for a file or set of files. As with the other LISTF command options, the fileset and listfile parameters remain the same. The behavior for these new options will be as similar to the MPE/iX options as is possible with respect to the operating system differences that exist.

Enhancement to ALTACCT, ALTGROUP, and ALTUSER

This enhancement provides for the use of +/- syntax in the "CAP=" specification of the ALTACCT, ALTGROUP, and ALTUSER commands. For example, to add PH capability and subtract MR and PM capability from user BOBH, the following command would be used:

```
ALTUSER BOBH;CAP=+PH,-MR,PM
```

Note that + or - starts an action of add or subtract that continues until the sign changes, so in the above case, "PM" is equivalent to "-PM".

This change preserves existing MPE capability rules. Examples are:

* An attempt to remove AM from an account is ignored.

* Removal of all capabilities from a group is not allowed and attempting this results in default group capabilities IA, BA.

* SM capability cannot be removed from the SYS account.

* UV cannot be removed from a user or account without also removing CV (this occurs automatically and a warning message is issued).

* Removal of both IA and BA from a user or account is not allowed; a warning is issued.

Introducing the CHGROUP Command on MPE V

In Release 30, the MPE/iX CHGROUP command is available on MPE V. This command allows you to change your logon group without logging off and back on again. This not only saves time, but also reduces system resource requirements by eliminating costly logoffs/logons.

The syntax for the CHGROUP command is the same as on MPE/iX:

```
CHGROUP [ [GROUP=]groupname][/grouppass]
```

If the groupname parameter is omitted, the user is switched back to the home group. If the password is not provided and the command is entered in a session, the user is prompted for the password. CHGROUP commands entered in a job must embed the password following the group name. Passwords are not required

if the target group is the user's home group. After switching the logon group, the entire Command Interpreter environment is preserved (for example: temporary files, file equations, cataloged UDCs).

This command is available in a session or a job, but not in break or from a program. Pressing BREAK has no effect on this command.

CAUTION:
As of Release 30, all unsupported PM utility programs used to switch the logon group should be removed. This is because few of these programs update the proper directory and job/session tables required to properly switch the logon group. When these programs are used with the MPE CHGROUP command, system failures (mainly SF406s) may occur upon issuing the CHGROUP command when the command executor finds an inconsistency in the associated directory entries. Furthermore, none of these utility programs can properly adjust the directory connect and CPU time counters when switching groups. The CHGROUP command will update these counters as necessary upon switching into a new group. Supported third party tools used to switch logon groups should be certified for use with MPE V Release 30 by contacting the software supplier.

Programmatic Execution of RUN and Subsystem Executors

Until now, subsystem executor commands (those MPE commands that create a process) have never been available through the COMMAND intrinsic (otherwise known as programmatic execution) on MPE V. On the other hand, these commands have always been available through the COMMAND intrinsic on MPE/iX. The availability of these commands through the COMMAND intrinsic provides many advantages:

1. Developers no longer need to use the process handling intrinsics CREATE, CREATEPROCESS, ACTIVATE, SUSPEND, etc. to RUN a program from another program. In Release 30, one can create a process programmatically by simply passing the "RUN PROGNAME..." string to the COMMAND intrinsic as can be done through the COMMAND or HPCICOMMAND intrinsics on MPE/iX.

2. Existing applications and MPE subsystems become more powerful. Most MPE subsystems and user applications allow MPE commands to be executed by prefacing the application command with a ":". In MPE V Release

30, these applications will now be more flexible with no code changes because users will now be able to RUN other programs without leaving the main application program or subsystem. For some customers, however, this could pose a security issue unless other precautions are taken. This will be discussed in more detail shortly.

3. By not requiring users to exit programs to run other programs, process creation rates will go down, thus reducing the demands on system resources.

4. The increased command interpreter flexibility will facilitate the creation of more powerful UDCs and job streams with less effort.

In addition to the existing MPE commands allowed through the COMMAND intrinsic, the following subsystem executor commands (those commands that create a son process) can now be executed programmatically:

RUN	RPG	SPL	FTNGO
FTN	FTNPREP	PREP	RPGPREP
SPLPREP	SEGMENTER	PREPRUN	RPGGO
SPLGO	VINIT	FCOPY	COBOL
DSCOPY	SYSDUMP	EXPLAIN	COBOLPREP
BBASIC	FULLBACKUP	EDITOR	COBOLGO
BBASICOMP	PARTBACKUP	PASCAL	COBOLII
BASICOMP	BASICPREP	PASCALPREP	COBOLIIPREP
BBASICPREP	BASICGO	PASCALGO	COBOLIIGO
BBASCIGO	FORTPREP	BASIC	FORTRAN
APL	FORTGO		

In order to execute a programmatic command from another program, either the user issuing the command must have PH capability or the program the user is issuing the programmatic MPE command from must have PH capability. These capability requirements match those employed on MPE/iX.

CAUTION:
Some MPE V customers enforce a security policy where all users are locked into a single application and allow MPE access only through the COMMAND intrinsic. Beginning with MPE V Release 30 and on MPE/iX, the users will now be able to RUN other programs from the main application, which may violate these customers' security policies. If the program does not have PH capability, this will not be a problem unless the user running the program has PH. However, if the program has PH capability, the system manager and application support teams must address this issue in one of two ways:

1. Use the Security Monitor product to disable programmatic access to subsystem executor commands. A potential drawback of this approach is that SM users will be the only users on the system able to execute subsystem executor commands programmatically.

2. Modify the application to filter out subsystem executor commands before calling the COMMAND intrinsic.

Escape Sequence Edit Changes for TELL, TELLOP, WARN

MPE has always edited messages sent to user terminals to make sure one could not embed escape sequences in messages that could do undesirable things to the target user terminals. An example of such an escape sequence is an [escf] which does a modem disconnect, or an [eschescJ] which does a home/clear on the target user terminals.

Unfortunately, stripping out escape sequences other than simple video alterations made it impossible to include other desirable escape sequences. For example, some third party terminals have a 25th line or message window that can be written to using a special escape sequence; however, MPE will not allow these escape sequences to pass through. Another example is if someone wants to intentionally do a home/clear on the target user terminals before sending a message.

This issue has been addressed in Release 30 by allowing users with SM or OP capability to send unedited messages via the TELL, TELLOP, and WARN commands. Users with these special capabilities must ensure that no unwanted escape sequences are embedded in the message. Spooler forms messages and console messages initiated by the PRINTOP and PRINTOPREPLY intrinsics also had the same escape sequence restrictions prior to Release 30. Beginning with this release, these messages also pass through unedited by MPE, provided the user has SM or OP capability.

TurboIMAGE/V Enhancements

Beginning with Release 30 of MPE V, TurboIMAGE allows search and sort items to be modified via the DBUPDATE intrinsic. The feature is referred to as "Critical Item Update" and has already been released in

MPE/iX 4.0. In all previous versions of TurboIMAGE (and IMAGE/3000 as well), DBUPDATE was limited to the modification of non-critical items. By way of definition, critical items are KEY items which are found in master sets, and SEARCH and SORT items which are found in Detail sets. DBUPDATE can now modify SEARCH and SORT items. Modifications of key items in a manual master dataset still requires a DBDELETE and DBPUT.

Since the primary intent of this paper is discussion of new features to the MPE V operating system itself, TurboIMAGE/V enhancements will not be discussed further here. Please refer to the Release 30 Communicator for further details.

MPE V Systems Now Recognize the Year 2000

In Release 30, changes have been made to MPE V to allow the system to recognize dates in the year 2000 and beyond. The general strategy for these changes are as follows:

* Where two digit dates are required, such as entering the date when starting the system, the digits 00 through 27 will represent those years in the 21st century (20xx). The digits 28 through 99 will represent those years in the 20th century (19xx). For example, if you enter the date 1/1/00 when starting the system, the system will start with the date January 1, 2000. Other examples would be where you are specifying date parameters in the STREAM or STORE commands or the date parameter in SYSDUMP.

* Where 4 digit dates are returned from the system, such as the DATELINE intrinsic, the date returned will have the year as either 19xx or 20xx as appropriate.

* Where "year of century" is returned by the system such as the CALENDAR intrinsic, or is asked for by the system such as the FMTDATE or FMTCALENDAR intrinsics, the "year of century" field [bits .(0:7)] is now defined as "# years since 1900". For example, if the value returned in this field by the CALENDAR intrinsic is 100, the current year would be the year 2000 (1900 + 100 = 2000).

It is suggested that you check all applications and programs for the correct handling of the year numbers 2000 and beyond. This testing can be done by doing a

WARMSTART on the system and entering a year in the 21st century (20 gives the year 2020 and is a leap year with the dates all the same as 1992. For example, 10/10/92 and 10/10/20 are both a Thursday).

CAUTION:
Do not do this when other users are on the system, because applications could get errors, may not run, or may produce incorrect data. Also, do not use the INTEREX Contributed Library program CLKPROG or any similar program to change the date and time with other users on the system. This could also result in incorrect data, including the system accounting information displayed with the REPORT command. If, for example, a user is logged onto the system and the system date is changed to the year 2020 and then the user logs off the system, the "connect time" that will be recorded for the user will be the total seconds as if the user had really been logged onto the system from 1992 to 2020.

MPE V Release 31

EDITOR File Security Enhancement

EDITOR A.08.00, releasing with MPE V Release 31, has been enhanced with a long-requested security feature - the optional ability to pass on security attributes of the TEXT file to the KEEP file. This is implemented through a new global option on the SET command. The syntax is:

```
/SET SECURE       << Activates KEEP file security.      >>

/VERIFY SECURE    << Display status of KEEP file        >>
                  << security as shown below.           >>

SECURE = TRUE (I.E. NOSECURE = FALSE)
                  << Indicates security is active.      >>

/SET NOSECURE     << Deactivates KEEP file security.    >>

/VERIFY SECURE    << Display status of KEEP file        >>
                  << security as shown below.           >>

NOSECURE = TRUE (I.E. SECURE = FALSE)
                  << Indicates security is not active.>>
```

The file security attributes that are mapped from the TEXT file to the KEEP file when SECURE is TRUE are:

* Security matrix

* Lockword

* SECURE / RELEASE status of file

* Access Control Definitions

Note that when SECURE is FALSE, the KEEP command behaves as it always has, creating a file with standard EDITOR new file default security. The default setting for SECURE upon first entering EDITOR is FALSE, thereby preserving EDITOR's original behavior. However, some users may wish to enter EDITOR with the initial value of SECURE set TRUE. Two methods are available for doing so: EDITOR can be run with PARM=1, or the JCW EDITORSETSECURE can be set to 1. Examples are shown below:

```
:RUN EDITOR.PUB.SYS;PARM=1
                    << Initial value of SECURE is TRUE. >>

:SETJCW EDITORSETSECURE=1
:EDITOR             << Initial value of SECURE is TRUE. >>

:DELETEVAR EDITORSETSECURE
                    << New on MPE V with Release 31.    >>
:EDITOR
                    << Initial value of SECURE is FALSE.>>
```

Regardless of the chosen initial value, once within EDITOR the user can control the setting of SECURE with SET as shown above.

Passing of security attributes from the TEXT file to the KEEP file occurs only when the following three conditions apply:

1. SECURE is TRUE. This must be done prior to the first KEEP command for which security is desired, and is described in detail above.

2. The most recently TEXTed file is a permanent disc file.

3. The KEEP file is a permanent disc file.

Note that when the above three conditions apply, KEEP will apply TEXT file security in all of the following cases:

1. When the KEEP file is a new permanent disc file.

2. When the KEEP file is an existing permanent disc file that is not the TEXT file.

3. When the KEEP file is the TEXT file. This allows preservation of existing security when modifying and KEEPing an existing permanent disc file.

The value of the SECURE option, TRUE or FALSE, remains in effect through the editing session until modified with SET or until EDITOR is exited and re-entered.

Note that EDITOR does NOT apply security to EDITOR work files ("K" files) that are created while SECURE is true, and does NOT apply security to workfiles that are renamed as a result of the KEEPQ command.

Spooler now Supports the MPE/iX SPSAVE Feature

The SPSAVE feature in the Native Mode Spooler allows one to save a copy of a spool file after the spool file has printed. This feature is now supported in the MPE V spooler beginning with Release 31.

The command interface is the same as MPE/iX. The command changes made to support SPSAVE are summarized below:

* The JOB command now supports the SPSAVE keyword which sets SPSAVE on the STDLIST spool file.

  ```
  !JOB REPORTS,MGR.FINANCE;OUTCLASS=PP,13;HIPRI;SPSAVE
  ```

* The FILE command now supports the SPSAVE keyword to set SPSAVE on a new spool file.

  ```
  :FILE PAYROLL;DEV=PP,13,2;ENV=LP602.HPENV.SYS;SPSAVE
  ```

* The LISTEQ command will show ";SPSAVE" if set on a spool file equation.

  ```
  :listeq

  FILE EQUATIONS

  FILE PAYROLL;DEV=PP,13,2;ENV=LP602.HPENV.SYS;SPSAVE
  ```

* The ALTSPOOLFILE command now supports the SPSAVE keyword to set SPSAVE on an old spool file in the READY, OPEN, ACTIVE, or LOCKED stated.

  :ALTSPOOLFILE #O123;SPSAVE

* The SHOWOUT command has been modified to SHOW SPSAVE if present. This was done by changing the "FRM" column that formerly showed "F" when a forms message was present to " FS" which now shows "F" under the "F" column for a forms message, and "S" under the "S" column to indicate SPSAVE.

The SPSAVE implementation on MPE V had to be slightly different when compared to MPE/iX because the Native Mode Spooler interface and features are different than the CM spooler on MPE V. Details on these differences are documented in the Release 31 Communicator, available September, 1993.

SPOOK5 has been modified to support SPSAVE; see the following section on SPOOK5 enhancements for details.

The following vendors have been notified of this enhancement and should be modifying their software to support SPSAVE on MPE V:

HOLLAND HOUSE's UNISPOOL
UNISON's SPOOLMATE
QUEST's NBSPOOL
NSD's TRANSPOOLER

MPE V will not support SPSAVE using RFA as is the case with the current version of MPE/iX. This means that a spool file created on a remote MPE V or MPE/iX machine using SPSAVE in the file equation (i.e. FILE OUT;DEV=NODENAME#PP;SPSAVE) will not have SPSAVE set on the target system. If SPSAVE needs to be set on a remote spool file, use one of the following two means:

1) Create the spool file locally with SPSAVE and let a third party networked spooler package that supports SPSAVE transfer the spool file to the remote system. The networked spooler application can choose to save the SPSAVE spool file copy on either the source or target system.

2) RFA can still be used to manually transfer a spool file, but to set SPSAVE, the ALTSPOOLFILE...;SPSAVE command will need to be issued against the spool file on the remote system once transferred.

SPOOK5 (SPooler lOOKup) Utility Enhancements

One of the most used MPE V utilities is SPOOK5 (so named because it can look at "ghost" spool files, i.e., spool files that cannot be seen with the MPE LISTF command). Many enhancement requests have been received for this utility; some of the major enhancements implemented in MPE V Release 31 are discussed below. For a full description of all enhancements, syntax, and fixes to SPOOK5, refer to the Release 31 Communicator.

New HELP facility:

A new HELP facility has been added that gives full syntax, parameter, and operation details with examples.

New REDO Command:

A new REDO command has been added. This command is especially handy if you have to re-issue a command (such as the ALTER command) several times with only minor changes, or if you enter a lengthy command (such as the DELETE command with a long list of Device File IDs) and make a mistake in the middle of the command. Only the last command is available to REDO; there is no "REDO stack".

Changes to the DELETE command:

The DELETE command has been changed so it can be called by just using "D" or any amount of characters that spell the DELETE command ("DE", "DEL", "DELE", or "DELETE"). To do this, the DEBUG command was changed so that it can only be called using "DEB", "DEBU", or "DEBUG".

Expanded MODE Command:

The MODE command has been expanded to allow the user to better control how certain actions will be accomplished during the entire time the program is being used. These items, however, will not carry forward to the next time you run the utility. Among the many new features of MODE is the ability to control whether the FIND command will stop upon the first character match, or continue on displaying all matches found. Also, the width of lines written to $STDLIST can be controlled, which is a very useful feature for terminals that can handle long line displays. The new SET command is exactly the same as MODE and is provided as a convenience to users more used to "SET" as a means of controlling parameters.

Command Input Length Increased:

The previous command length limit of 80 characters has been upped to 276 characters. This is especially useful for OUTPUT or INPUT commands which could easily exceed the old limit. This also facilitates the ability to issue lengthy MPE commands from within the utility.

List without line numbers:

There are now two methods of listing the contents of a spool file without the line numbers. The first method is to specify ",UNN" in the LIST command and the second method is to specify "QUIET=ON" in the MODE or SET command.

Setting SPSAVE status for an output spool file:

As of Release 31, MPE V has an enhancement allowing you to specify that an output spool file is to be created with SPSAVE status so that after the last copy of the file has been printed, MPE will reset the priority of the spool file to 0 and save the file instead of deleting it. With this enhancement, you can now set the SPSAVE status for output spool files with the ALTER command. Further, the output of the SHOW command has been modified so that if the file has SPSAVE status set, it will show an "S" in the "RFS" column of the file detail listing. For further information, use the HELP command to refer to the PARMS, OPERATION, and EXAMPLE of the SHOW command.

NLS Compatibility:

The utility has been enhanced to allow localization of the message catalog file. To support the message catalog, the LANG= parameter of the MODE or SET command has been implemented in this version. The following files on the PUB.SYS account are used:

SPCAT000 - GENCAT prepared default message catalog.

SPCATSRC - Source of the default message catalog that can be converted to any language.

SPHLP000 - MSGCAT prepared default HELP file.

SPHLPSRC - Source of the default HELP file.

Protection against accidental purging of MPE files:

The SPOOK5 DELETE command will accept wildcards to delete spool files, while the SPOOK5 PURGE command will only accept dfid (Device File ID) numbers to purge spool files. However, with the enhancement to the MPE PURGE command to accept wildcards, it has been found that some people will accidentally purge MPE files because they forget which SPOOK5 command will accept wildcards and try to PURGE @.@ spool files. If "PURGE @.@" were entered within SPOOK5, the command would be passed to MPE for execution. While MPE states the number of files that will be purged and asks the user if they want to continue, it has been found that the users still thought that the question was asking about spool files, not MPE files. As a result, users would respond "YES" and would purge all files of all groups of the logon account if they had AM (Account Manager) capability.

To help keep this from occurring in session mode, two things have been done:

1. A change was made to the MPE message clearly stating that the number of files selected are "MPE FILES".

2. SPOOK5 has been changed to first give the user a warning that the utility could not execute the command and it would be passed to MPE for execution. Then, the user is asked if they want the command to be passed to MPE for execution. If the user enters "NO" or simply presses the carriage return key, the command will not be passed to MPE and the user will get back to the SPOOK5 prompt.

With two full opportunities to stop the command, first by responding to a question from SPOOK5 and then by responding to a question from MPE, it is felt that the accidental purging of MPE files should only occur in rare circumstances.

In batch mode, neither SPOOK5 nor MPE will ask any questions. If SPOOK5 cannot execute the command in batch mode, it will automatically pass the command to MPE to be executed. In MPE, when executing the MPE wildcard PURGE command, the user is not asked to verify that they want this command executed.

JCWs can now be Removed with the DELETEVAR Command

Prior to Release 31, a job/session had no way of removing JCW entries other than logging off and back on again, which would remove all JCWs. On MPE/iX, CM JCWs can be removed with the DELETEVAR command. The DELETEVAR command also deletes native mode CI variables on MPE/iX.

Beginning with Release 31, the DELETEVAR command is available on MPE V to remove specific JCWs or all JCWs for a particular job/session. Note, however, that this command does not work with CI variables on MPE V because CI variables are not available on MPE V. The DELETEVAR name, however, was selected for removing JCWs on MPE V to provide compatibility with the existing command that performs this function on MPE/iX.

DELETEVAR Syntax

SYNTAX

DELETEVAR jcwname

PARAMETERS

jcwname	The name of a valid job control word (JCW) or @ which indicates all user-defined JCWs.

DELETEVAR may be issued from a session, job, in BREAK, or from a program. The command is not breakable. System-defined and system-reserved JCWs cannot be deleted.

Introducing the HPDEVCONTROL Intrinsic on MPE V

The MPE/iX HPDEVCONTROL intrinsic is now available on MPE V. This intrinsic provides access to specified peripheral functionality without the device being opened. The functions currently supported include remote tape loads and remote tape online for 7980 and DAT tape devices. This facilitates unattended backup and tape verification operations. The remote DAT online is also useful to place the DAT drive online without having to eject and reload the tape. The supported functions, calling sequence, parameter types, and error returns match the HPDEVCONTROL intrinsic on MPE/iX. This allows one to run an HPDEVCONTROL program on MPE/iX that was developed for MPE V with no code changes.

HPDEVCONTROL Calling Sequence

```
                        D      BA      DV      D
    HPDEVCONTROL( status,ldev,itemnum, item);
```

Operation Notes

This intrinsic is only supported on tape devices (type 24) whose subtype is 5 (HP7980) or 6 (HPC1511A - DDS). This intrinsic cannot be called in split-stack mode. If the device is not in an available state, HPDEVCONTROL will block until the operation can be performed on the device.

INITIAL Checks Free Space on LDEV 1 During a Coldload

Prior to Release 31, a customer could be forced into an unplanned reload if a coldload operation failed because of insufficient contiguous disc space for the system SL on tape, or if there was insufficient space for INITIAL to allocate certain system tables on disc. Beginning with this Release, SYSDUMP and INITIAL have been enhanced to prevent this problem from occurring. The remainder of this section describes the problem addressed by this enhancement and the method used in this release to overcome it.

Inadequate free disc space on ldev 1 can cause INITIAL to halt in one of two ways - both lead to an unplanned reload:

1. A patch, user software, and/or third party software installations adds new or increases the size of existing segments in SL.PUB.SYS such that a new extent in the SL is allocated and the size of the SL file increases accordingly on the coldload tape. This tape is then used to coldload the same or a different system that has little free space on ldev 1 - i.e. there is not a contiguous free block of space on ldev 1 large enough for the new SL file on tape. Note that if the SL file on tape is bigger than the old SL file on disc (as will be the case in the scenario described in the prior paragraph), the free space created when INITIAL purges the old SL will not be enough to accommodate the new SL file on tape.

If this happens and free space cannot be found elsewhere on ldev 1, INITIAL halts with the following error

ERROR 326 - OUT OF DISC SPACE ON LDEV 1

When this happens, INITIAL is too far into the coldload process to perform a start from disc, which results in a reload situation.

2. Even if the size of SL.PUB.SYS does not increase, the ERROR 326 can still occur if free space is extremely low on ldev 1. This is because INITIAL needs some contiguous free space on ldev 1 to copy system tables to disc during the coldload process. The minimum amount is around 2500 contiguous sectors. If less than 2500 sectors of free space exist on ldev 1, the ERROR 326 can occur, forcing a reload regardless of whether or not the size of the system SL file on tape has increased.

Problem 1) has been addressed as follows: When SYSDUMP creates a coldload tape, it will now write the size of the new system SL in the SYSDUMP/INITIAL communication record. During an UPDATE or COLDSTART, INITIAL will now compare the size of the new SL on tape with the size of the SL on disc. If the SL on tape is larger than the SL on disc, INITIAL attempts to allocate a large enough block of disc space BEFORE purging the old SL. If sufficient space cannot be allocated, the operator will be prompted to make a decision. The operator can either proceed with the coldload at the risk of a forced reload, or can stop the coldload at this point, restart from disc, clean up disc space, and attempt the coldload again later.

Problem 2) has been addressed by modifying INITIAL to always attempt pre-allocation of 2500 contiguous sectors. This is done early enough in the COLDLOAD process so that a start from disc can be done if the allocation fails.

Final Update on Year 2000 Changes for MPE V

Most of the changes required to make the MPE V systems recognize dates in the year 2000 and beyond were completed in Release 30. Two functions, however, remained to be certified to work in the year 2000. These functions were labeled tapes and SADUTIL (Stand Alone Disc UTILity). As of Release 31, both functions

are now certified to work in the year 2000 and beyond. Labeled tapes did not require any changes and will actually work in the year 2000 as of Release 30. SADUTIL, however, did require changes and only the Release 31 or later version (3.16) will work in the year 2000. Please be sure to create a new Stand Alone Diagnostic tape for your system when updating to MPE V Release 31 or later. Instructions are in Chapter 11 (SADUTIL) of the Utilities Manual (32033-90008) or in your MIT Update documentation.

Thanks to Steve Smead, Yukihiko Umezawa, Len Croley, Dan Clavin, Bob Holdsworth, John Green, Robert Ross, and others at HP SWT Division for their engineering efforts on MPE V enhancements and for providing material for this paper.

Centrally Managing Your Enterprise with HP OpenView System Manager

by Diane M. Bassett
Hewlett-Packard Company

Introduction and key issues

Managing your enterprise in this day and age is no simple trick. In fact, two out of three of Gartner Group's Key Management Issues for 1993 center on this topic ("What IS strategies can best balance the needs of end users and enterprise manageability?" and "How can IS meet increasing support demands with limited resources?"). Information Systems departments are feeling the crunch of the following issues:

- Staffing costs are high when staff must be trained and available in each location
- Pressure to "do more with less staff". In many shops, staff headcount is shrinking, and the company may be considering outsourcing operations. At the very least, headcount is not growing. But demands on IS *are* growing. How can IS departments meet user needs, provide a competitive edge to the company, and salvage their own jobs?
- Information resources may need to be distributed geographically in order to better support the business. But the *smooth management* of those resources is also critical in supporting the business. How can one succeed at both?

Centrally managing your enterprise

Centrally managing your enterprise, and automating as many functions as possible, is an effective way to reduce operations costs while maintaining or even increasing the level of services you can provide.

If you aren't able to increase your staff, you need to make your existing staff more productive. You can centralize your highly trained operations staff and leverage their skill set over a larger number of systems (because they don't have to be physically present at each system in order to manage it). Some customers have completely eliminated any IS staff at remote locations, leaving office personnel to perform simple tasks such as replacing a DDS tape in the drive. Centralizing your operations expertise reduces training costs (not as many people to train), and the opportunity for human error and inconsistent procedures.

> "One of the big benefits for us was the ability to have the remote systems management. We have had HP 3000s in our production facilities, but we also have a group of people that take care of running them that are very technically qualified. We don't have those type of individuals at our customer service facilities, so the ability to have remote system management was very important. Because as the this system rolled out to nine new locations we were able to deal very effecitvely from a cost standpoint with the system management without having to worry about having that technical expertise in these new locations."
>
> *Ken Thome,*
> *VP and Director Information Systems*
> *General Mills*

If you want to leverage your operations staff to manage a larger distributed number of systems, you must also provide them with tools to make them successful. Managing systems "by exception" is one important way to do that. This means that rather than monitoring all the messages that come across the console, the operator is only alerted to specific messages that indicate something has gone wrong, or requires their attention. It is as though the system is sayiing "everything is going well unless I tell you otherwise". This enables your staff to successfully manage a larger number of systems because the number of messages they have to monitor and respond to has decreased.

Another way to improve your staff's productivity (and reduce costs) is to automate as many functions as possible that previously required human intervention. Reducing the amount of time your staff spends responding to messages by typing on the keyboard will increase the amount of time they can spend on other projects.

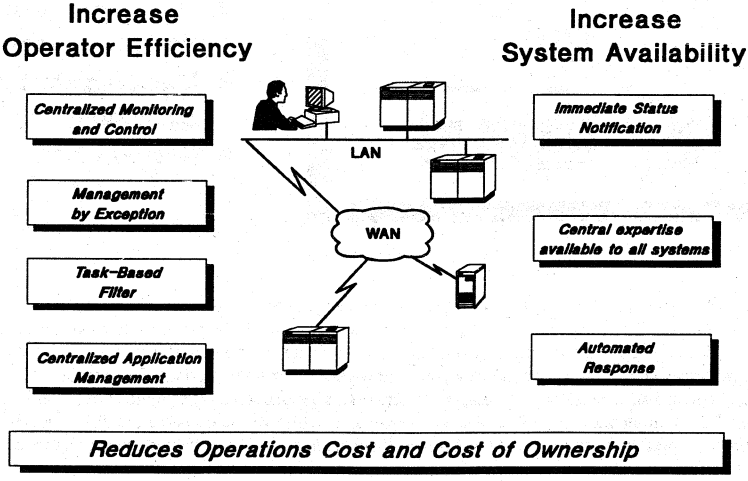
Automating can reduce costs to the point where outsourcing information technology functions may not be necessary. Outsourcing is on the rise in Europe. Gartner Group survey shows that 71% of CIOs said they were ready to outsource some operations in 1992, compared with 36% in 1991. The high costs and unavailability of certain staff are sited as a main reason for outsourcing. Automating your operations may enable you to salvage jobs for your staff, while making them more productive and valuable to the company.

What is HP OpenView System Manager?

HP OpenView System Manager is a client/server software product that runs partially on an HP 3000 host, and partially on a DOS PC with Windows. It allows operators to monitor and control multiple HP 3000s from a central (personal computer) console.

It consists of three main pieces:

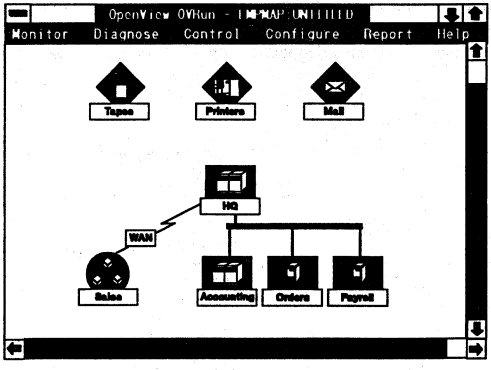
- HP OpenView Console, software which serves as the headquarters, the managing node for the other nodes. It runs on the host HP 3000, and also provides one set of PC floppy disks that run on the PC.
- A DOS PC with HP OpenView Windows and some other software to serve as the client. This PC can be ordered piece-meal, or as a bundle from HP.
- Remote node licenses. These enable the OpenView Console to manage up to 64 remote nodes, anywhere in the world.



Windows-based interface and management by exception

These pieces work together to give you an easy to use, windows-based console that alerts you when things go wrong. Rather than having to monitor the messages that come by on the screen, you are only asked to pay attention to exception conditions (thus the term "managing by exception"). The figure below illustrates the user interface:

Easy to use interface

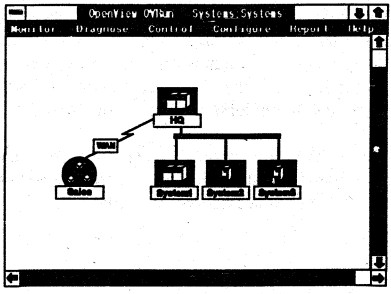


- **Windows-based**
- **Icons change color**
- **No lost messages**
- **Management by exception**
- **Nested maps**

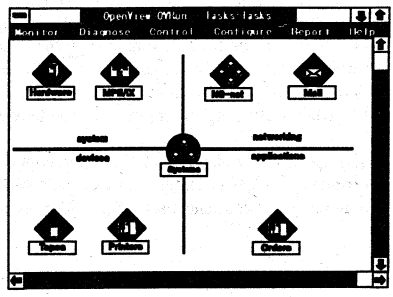
The console screen uses icons that represent various parts of the enterprise. For example, a disk icon can represent all the disks on the network. A printer icon can represent all the printers. Each system and/or application can be represented with an individual icon. How the icons are used is up to you and allows you to map your console screen to the way you want to think of your enterprise. For example, some people like to see all the disk activity across the network under one icon. (The figure above is an example of this.) Others might want to think of disk activity in association with the system the disk is attached to. An icon can represent an entire remote datacenter, that expands to show icons underneath that represent individual systems.

"Nested maps" is the name for the feature just described, where clicking on an icon reveals a map of other icons underneath. Multiple layers are possible. If an icon resides inside a circle, it means there are more maps and icons underneath it. If it resides in a parallelogram, there aren't maps and icons underneath. In the figures below, the map on the left is the "top level" map, and although the color doesn't show here, the round figure on the left represents a sales system that's turned red and has some sort of problem. By clicking on the red icon, the operator then sees the map shown on the right, which explodes the detailed icons associated with the sales system (note how the sales system is now in the center). If we could include color in this paper, you'd see that the icon for the tape drives has turned red, and this is the event that triggered the sales system to turn red in the top level map.

Top Level Map:



Secondary Level Map:



The benefit of nested maps and having a flexible way to look at the enterprise is that it enables you to use the product however you want to mentally think of your enterprise. It doesn't force a certain map structure on you.

The "management by exception" feature alerts you to a situation requiring your attention by changing the color of the icon. Green icons mean everything is working well. A yellow icon means that an event has occured that deserves attention. A red icon is more severe, and is assigned to events that are deemed serious and in need of immediately attention. For example, if the disk icon turned red it would indicate that a disk was having a problem. Linking events like this to their appropriate icons is called "task based filtering of events".

No more lost messages

With traditional consoles, operators run the risk of not noticing a message that requires a reply, or some action on their part. For example, a user could send a message requesting a particular tape to be mounted, and if a lot of messages are scrolling up the screen, it's possible for the operator to miss the request entirely. With OpenView System Manager, a request to the operator could result in a designated icon changing from green to red. The message is captured and is easily found by the operator. This feature then increases end-user satisfaction with MIS responsiveness.

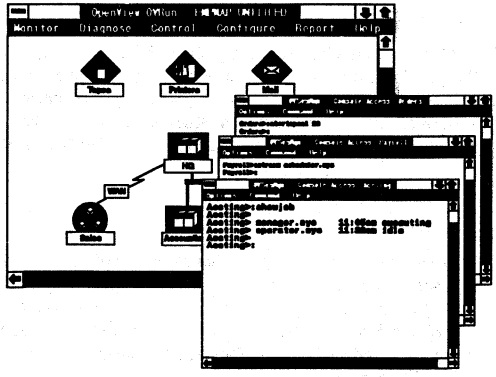
HP OpenView System Manager can be made to interface with applications that provide telephony services. For example, your operator could wear a beeper that alerts the operator to a red-icon event. This would enable the operator to leave the console and work throughout the datacenter or site. One such telephony product is Watchman, by Unison.

Easy to use interface

It has already been mentioned that the PC used to control the enterprise is windows-based, which makes it easy and intuitive to use. In addition, enhancements are provided that improve the way that operators look at events and messages. When, for example, an icon changes from green to red, the operator simply clicks on the icon to see a summary of the messages associated with that icon. The detail is suppressed at this point. By merely clicking on a specific message, full detail of that event is provided, as well as an opportunity to annotate any action taken (or instructions for the next shift). This can eliminate the need to have a written log for communication between shifts while still providing a way to "trouble track".

The operator can also have full console control of any HP 3000 in the network (including shutdown and restart) by simply clicking open a console window for a designated system. Multiple console windows may be open simultaneously. The figure below illustrates what console windows look like:

Full Console Access for System Control



Control ...

- **System Shutdown (Control-A)**
- **System Restart (Control-B)**
- **MPE Command Execution (ex. restarting network)**
- **Applications**

★ *Automated Response for Timely Problem Resolution*

Automated responses to events

In order to reduce operating costs, it's important to automate as many functions as possible. HP OpenView System Manager can use MPE scripts to provide an automated response to a particular event. A script is a small string of commands, written as though an operator were typing them in. In anticipation of typical tasks, scripts can be created to execute automatically if a particular event occurs.

A good example of this involves network problems. Many IS departments have a staff member who is most knowledgable about networks, a network administrator. During normal operations, the network

may fail and the typical response is to simply restart it. Usually, if that fails several times, the system operator will then resort to calling the network administrator.

One could automate this process by providing a script that essentially says, "If the network fails, use this script to restart it. If it fails again, restart it a second time. If it fails again, contact the system operator or network administrator." In a recent Gartner Group survey, the average number of users per LAN administrator was 115-to-1. It's easy to see how using scripts to automate responses would increase the productivity of someone like the LAN administrator, whose resources are already stretched thin.

Automated responses are both flexible and sophisticated. One can instruct the system to try several different automated responses in a certain order before the operator is called to intervene. Automated responses maximize the number of functions that can be automated in the datacenter, removing the need for human intervention wherever possible.

Increasing uptime with automated responses.

Applications that improve system and data availability can be automated using HP OpenView System Manager. For example, HP SPU Switchover/iX is a product that provides warm standby in the event of a system failure. The switchover function is easily automated with a simple MPE script. In this example, if System A fails, a notification is sent to HP OpenView System Manager and the automatic response to switch over to System B is given. Without this, the switchover application is dependent upon operator intervention. Automating the procedure will result in an immediate "switchover" response and increased uptime for end users.

Automated responses can also help out in the event of a data center disaster. Combining HP OpenView System Manager with NetBase results in a disaster tolerant, geographically dispersed HP 3000 "cluster" known as SharePlex/iX. NetBase is a software product by QUEST (also available from HP) that, among other things, shadows data from one location to another. Applications needing that data go through a directory that knows where the data resides. In the event of a disaster such as an earthquake or a bombing, NetBase will send a message to an OpenView Console (in another location) that is centrally managing the enterprise. An automated response could provide an alternate directory to redirect applications and users to find the data that is unavailable in the alternate (shadowed) location. This would enable the rest of the business to continue functioning, and would alert the central operations staff that an emergency had occurred..

Manage your enterprise with multiple consoles

The HP 3000 that is the central management node (the OpenView Console) can send information out to multiple PCs at the same time. This type of configuration is very helpful to IS departments with "lights out" or "lights dim" environments, and for very large data centers. Each PC (five is the maximum) could have the same "view" with the same icons, and could be located in different areas of the data center. Or, each PC could "specialize" such that one would handle all the messages and icons associated with tape drives, another PC would receive all the messages for disks and printers, etc. One staff member could monitor just the icons associated with applications.

Summary of benefits of central enterprise management

Centrally managing your enterprise with HP OpenView System Manager or a similar product delivers the following benefits:

Reduces cost of operations

- Staff productivity increases by using a tool that enables them to manage by exception.
- Training costs decrease because expertise can be pooled and applied to multiple, even geographically dispersed systems.
- Automated responses enable more "lights out" operations and less human intervention.

Increases system uptime

- Provides faster fault detection by notifying operators of the status of all networked HP 3000s.
- Automated responses enable correction to take place without waiting for an operator to become aware of the problem.
- Other applications can interface with HP OpenView System Manager to automate high-availability features such as SPU switchover, and SharePlex/iX disaster tolerant clustering functionality.

STREAMX as a Second Language

Paper No: 5039

By: Scott DeChant

MEDSTAT Systems
777 E. Eisenhower Pkwy, Suite 500
Ann Arbor, MI 48108
(313) 996-1180

Introduction

Popeye the Sailor used to say "I y'am what I y'am and that's all what I y'am." STREAMX, VESOFT's interactive, intelligent batch job creation facility, tends to say the same thing (albeit much more grammatically). Why should this be the case? There's no reason that a system shouldn't be stretched to its limits (whatever those may be), and that newer features and capabilities shouldn't be found and shared.

At MEDSTAT Systems, we have done a lot of research with STREAMX, having pushed, twisted, and "opened up the throttle" on it to see what it can do. Consequently, we have found some great uses for STREAMX. We have used STREAMX as a structured language of its own, increasing our capacity and giving us a larger tactical arsenal with which to surgically attack our customers' needs.

The purpose of this paper is to share some of these uses with other interested STREAMX programmers and users. Some of the concepts and accompanying examples are pretty basic in nature. The standard (and far too limited) documentation provided by VESOFT contains more basic examples. Other examples in this paper are a bit more complex, and a few push the STREAMX envelope.

This paper contains three sections describing possible STREAMX uses, ranging from basic principles to more advanced ones. Following these sections are two advanced program examples using STREAMX.

Basic Programming Techniques

Variable Prompting and Assignment

Variable prompting and assignment of values is at the heart of STREAMX as a programming language. STREAMX has variable space limits, depending on the number, and types of variables used as well as the HP machine running STREAMX (consult the published STREAMX information for details). However, variable use can be powerful.

STREAMX jobs work on two levels; the batch job "setup" level, and the batch job "run time" level. "Setup" refers to the time when the user is answering prompts and creating a job. "Run time" refers to the point at which the job is actually executing on the HP. Setup commands are preceded by a "::", while run time commands are preceded by a "!" or ":".

Below are 4 ways to assign values to variables.

1. **::ASSIGN < variable > = < value >**
   Assigns a value to a variable. The variable can be reset later. This command can only be used at job setup time.

   Example
   ```
   ::ASSIGN TEMPVAR = 1
   ```

2. **::PROMPT <var type> <var name> < prompt >**
   Prompts the user for input and stores the user response in a variable. This command can only be used at job setup time, and the variable can be reset later. The variable types allowed are string (STR), integer (INTEGER), and date (DATE).

   Example
   ```
   ::PROMPT STR TEMPSTR = "Name of file to be acted upon"
   ::PROMPT INTEGER TEMPI = "Number of files to be acted upon"
   ::PROMPT DATE DOB = "Date of Birth"
   ```

3. **:SETJCW < var name > = < int value >**
   Sets an HP Job Control Word to a specified integer value. This command can be used either at job setup time or at job run time.

   Example
   ```
   :SETJCW TEMPI =  1
   ::SETJCW STEP = 15
   ```

4. :SETVAR < var name > < value >
   Sets an HP variable to a specified integer or string value. This command can only be used at job run time.

   Example
```
:SETVAR STEP 1
:SETVAR NAME "Ted"
```

In STREAMX, all variables are global. So it is important to remember which variable names have already been used, or risk overwriting values. Programmers can take advantage of the global nature of variables by using the same names for temporary variables throughout a job. One instance of this is in asking a standard yes/no question. Usually the user provides an answer, something is done, and the variable isn't needed any longer. Using the variable YN for all such situations saves valuable variable space, and provides consistency in STREAMX programs.

STREAMX Variable Usage

Now that our STREAMX variables have values, how can we use these variables in our jobs?

1. **Prompt time input validation**
   Input validation is probably the most common form of error checking. In many cases, the programmer will have some idea of expected values for input to a set of logic. Programs can validate input at prompt time with the ::PROMPT statement as shown below.

   Examples
```
::PROMPT STR FILENAME = "File name"; CHECK = (FEXISTS(FILENAME))
::PROMPT STR YN = "Yes or No"; CHECK = (YN = "YES" OR YN = "NO")
::PROMPT INTEGER INT = "Enter a number 1 - 5"; &
::  CHECK = (INT >= 1 AND INT <= 5)
```

2. **Conditional logic**
   Another very common use of variables is for IF-THEN-ELSE conditional logic. Programmers can use this logic to point to other sections of logic, either to modify files or to alert the user to input errors.

   Example
```
::PROMPT INTEGER INT = "Enter a number 1 - 5"
::IF INT < 1 OR INT > 5 THEN
   :: ECHO Value not 1-5. Please consult the users guide.
::ELSE
   :: PROMPT STR FILENAME = "File name"
::ENDIF
```

3. **Value substitution**
   STREAMX can also be used to incorporate stored variable values into batch jobs. This is helpful when sending messages to users, as well as creating counters, parameterizing jobs and programs, etc. By putting the variable name in brackets the variable's current value is inserted into the job stream instead of the variable name itself.

   Example to purge files LOG1, LOG2, LOG3, LOG4 in PUB.SYS

```
::ASSIGN STEP = 1
::WHILE STEP <= 4
:: ASSIGN   FILENAME = "LOG{STEP}.PUB.SYS"
:: ECHO This  is step # {STEP}
:: PURGE {FILENAME}
:: ASSIGN  STEP = STEP + 1
::ENDWHILE
```

4. **Use with functions**
   VESOFT provides many functions to complement HP system functions. String variables can be up-shifted, parsed, "substringed," and compared, while numeric variables can be part of arithmetic expressions. Most functions needed exist either from HP or VESOFT. Consult the MPE/XL help facility for information on the HP functions as well as the VESOFT functions.

   Examples

```
::ASSIGN YN = UPS(YN)
::SETJCW INT = I + J
::ASSIGN SUBSTR = STRING[0:5]
```

Intermediate Programming Techniques

Looping and Multiple File Usage

Another feature you can take advantage of is the ability to use a variable as an index for another variable. Strictly speaking, STREAMX does not support arrays, but arrays can be "fudged" by appending the "array index" to the end of the variable name (i.e., instead of using VAR(1), VAR(2), we use VAR1, VAR2 as the variable names).

Sometimes, we don't know how many input files will feed into a process. We only know that we must use all of them, merge them into a larger file, and then use the merged file as input to the process. The "pseudo-array" approach lends itself well to this construct.

Example:

```
::PROMPT INTEGER NUMFILES = &
:: "Number of rawdata files to run through program A"
::ASSIGN TEMPSTR = "INPUT SORTFIL1"
::ASSIGN I = 1
::WHILE I <= NUMFILES
:: ECHO Rawdata file #{I}:
:: PROMPT STR RAWDATA  =  "Rawdata file name"
::
:  COMMENT  Sort raw data file
:  RUN SORT.PUB.SYS
       INPUT {RAWDATA}
       OUTPUT  SORTFIL{I}
       KEY 1,10
       END
:
:: IF I >= 2 THEN
::     ASSIGN TEMPSTR = TEMPSTR + ", SORTFIL{I}"
:: ENDIF
::
:: ASSIGN  I = I + 1
::ENDWHILE
:
: COMMENT Merge files together
: RUN MERGE.PUB.SYS
   {TEMPSTR}
   OUTPUT MERGDATA
   KEY 1,10
   END
:
:  COMMENT Run program A
:  RUN PROGA
       MERGDATA
       OUTPUT
:
```

Note that each array element is technically a new variable, so variable space is used up much more quickly than if the same variable was used over each time. Use the same variable name whenever possible, but if you need to save the names of all of your related elements, nothing is wrong with the "array" approach if you have enough memory space.

Object-Oriented Design Applicability

Just as with other structured programming languages, STREAMX can lend itself to object oriented design. This is not object-oriented implementation (a la C++, Smalltalk, etc.), but is a modular design that allows reuse of code and global modifications to code, reducing maintenance effort. STREAMX doesn't specifically support subroutines. Through the use of ::USE files, you can include code from other places, just as if it were in a subroutine. To

do this, however, you must keep in mind that STREAMX considers all variables global, so you must use the same name for variables in both the main program and in the subroutine (no argument passing allowed).

We've used this approach in three distinct ways at MEDSTAT Systems. We have developed STREAMX ::USE files to hold the prompts for a given program. This allows us to set up a generic overriding "umbrella" program to run all of our programs. We've also been able to set up menu-driven jobs that call other routines. This way we can build 30-50 step jobs using the same types of functions over and over, with different raw data files, and do it while only maintaining one "umbrella" program and a series of smaller routines. Therefore, we have little or no code duplication. Finally, we have combined STREAMX with HP command files to limit code duplication. These uses are detailed below:

Program Prompt ::USE Files

When creating generic programs or jobs, modular code is typically easier to use and maintain. One way to achieve additional modularity is to place all program prompts in an external ::USE file that is called by the main job. This way, if prompts change in a program, all you have to do is change the external file, instead of an entire job. This also alleviates the problem most generic "umbrella" jobs run into; how to handle a different number of prompts in different programs of the same type.

For example, if program A prompts for just input data and output data, while program B prompts for both of those plus a couple of numeric values, these programs could not be run by a single job without a lot of extra logic ("if program = B then prompt for numeric values," etc.). Notice the example below:

Overall STREAMX job

```
::
: RUN   ?Program  name to run?.EXE.PROD
:: USE ?Prompt ::USE file?.INCLUDE.PROD
::
```

Prompt ::USE file if running program A with above job

```
::
   ?Name of raw data?
   ?Output file name?
::
```

Prompt ::USE file if running program B with above job

```
::
   ?Name of raw data?
   ?Output file name?
   3
   5
::
```

Both programs can now use the same job to run them; it's just the ::USE files that change. You have now created modular code, and ideally only need one STREAMX job ever again.

Menu-driven Programs

Menu-driven programs can reduce the need for multiple programs or for extensive customization in programs. Generic code reduces maintenance and simplifies code management.

A simple menu-driven utility could look like this:

```
::ASSIGN OPTION = " "
::PROMPT STR INFILE = "Name of file to be acted upon"
::WHILE OPTION <> '//"
:: ECHO  Ways to modify files:
:: ECHO
:: ECHO  SO =  Sort   {INFILE}  RU Purge  {INFILE}
:: ECHO  CO  =  Copy   {INFILE} RN Rename {INFILE}
:: ECHO
::
:: PROMPT STR OPTION = "Step to do (RN etc.,  // to exit)";
::            CHECK  = (len(OPTION) = 2)
::ENDWHILE
::
```

At this point, the user can select a step (e.g. SORT or PURGE), and can be directed to the correct STREAMX subroutine:

```
::IF  UPS(OPTION)  =  "SO"  THEN
:: USE SORT.JOBSUB.STREAMX
::ENDIF
::
::IF UPS(OPTION) = "PU" THEN
:: USE PURGE.JOBSUB.STREAMX
::ENDIF
```

The sort subroutine could then look like this:

```
: COMMENT ---------- SORT FILE ----------------------------
::ECHO    ---------- SORT FILE ----------------------------
: RUN SORT.PUB.SYS
   INPUT {INFILE}
   OUTPUT ?Full name of sorted output file?
   KEY ?Sort key?
   END
```

You can now include more options as needed. Whenever you add a new option, just add it to the menu program and create the new subroutine. You may notice the use of INFILE as a standard variable name. Because of the global nature of STREAMX variables, establishing some common variable names as "reserved variables" will make using subroutines easier.

Combining STREAMX and HP Command Files

Sometimes you may want to use a certain logic interactively; other times you may want to use the same functionality in a batch-mode job. By combining HP command files and STREAMX, you can do both without duplicating code. The process involves writing a HP command file with all the parameters needed, then creating a STREAMX file that prompts for the appropriate parameter information and calls the command file.

Example HP command file PRGRP.CMD.SYS

```
PARM GROUP = " "
COMMENT PRGRP: Print all files in a group to the line printer
FILE LP;DEV=LP;CCTL
ECHO
RUN MPEX.PUB.VESOFT;PARM=1;INFO= "PRINT @.!GROUP;OUT=*LP"
```

Example of STREAMX code calling the above command file

```
::
::PROMPT STR GROUP = "Group to print reports from"
::ECHO Printing all reports in {GROUP} group of {HPACCOUNT}
::PRGRP.CMD.SYS {GROUP}
::
```

Note that you can call a command file from either STREAMX interactive mode or job run-time mode.

Advanced Programming Techniques

Self-documenting Job Streams

After making your programs generic and modular in nature, adding a feature such as self documentation is relatively easy. There are two ways to accomplish this. The first involves writing all steps and pertinent information to an external file as the program or job runs, thus creating an accurate report of what was done. The second is to create the documentation of steps and appropriate information up front, and have the documentation control what the program does.

Post-execution Reporting

This is simply writing information to a file. The file must be opened at the beginning of execution, and as each successive step is executed, more information is written to the file. One such example file is shown below:

```
THU,  MAY 13,1993,3:13 PM Starting    Template          Clock= 3
THU,  MAY 13,1993,3:13 PM Starting    Step 3   File merge Clock= 3
THU,  MAY 13,1993,3:19 PM Completed   Step 3   File merge Clock= 11
THU,  MAY 13,1993,3:19 PM Starting    Step 4   Data  Sort Clock= 11
THU,  MAY 13,1993,3:20 PM Completed   Step 4   Data  Sort Clock= 17
```

Important information in this case includes the date, time, step description, and information on CPU usage. Since this report is being created as the job is executing, the user can tell exactly where the process is in relation to completion, and can abort the job if something goes wrong.

Program-control Documentation

Another way to integrate documentation with execution is by using the documentation to set up the job. This is very similar to the menuing option. Instead of entering information from the keyboard, the user can set up a documentation file that the STREAMX program can read when setting up the job. One example of a tabular setup file is below:

```
Widgets 'R Us    1Q92 UPDATE      UPDATJOB  Keith Raisch
*
PRECONVERT       PROGA            RAWDAT1   KSAM preconvert
KSAM BUILD       KSAMUTIL         OUTPUT1   Create KSAM file
SORT             SORT.PUB.SYS     OUTPUT1   Sort the rawdata file
PURGE                             OUTPUT1   Don't need output any more
RENAME                            SORTDAT1  Rename sort data for storage
```

In this case, we've identified the company, process description, job name, and operator name at the top. In the heart of the file, we have the step name, the program to run, the rawdata for the program or step, and a comment for job documentation.

By keeping the setup table as a stored file, you have documentation of the process handy, and any changes needed for subsequent runs -- more programs/steps, different rawdata files, etc., -- can be handled easily.

By using both methods of documentation, you can have the best of both worlds, and can compare what was supposed to happen in a process to what really did happen during process execution.

Creating Accurate Test Environments

When creating a test environment for new programs, data, or even systems, it is important to mimic as closely as possible the actual production environment. The closer your test cases are to actual situations, the less likely it is that unexpected problems will arise. STREAMX is a great tool for simulating "battlefield conditions." With STREAMX you can test programs under different conditions by just changing a few variables.

One accurate way to create a test environment is by inserting a file containing only file equations for testing purposes at the beginning of your job. This is done with a ::USE file created for testing. You can build the testing logic into your job like this:

```
::PROMPT STR TEST = "Test run or production run (T/P) [P]"; &
:: DEFAULT= "P"; CHECK = (UPS(TEST) = "T" OR UPS(TEST) = "P")
::IF UPS(TEST) = "T" THEN
:: USE ?Full name of testing file equations file?
::ENDIF
::
```

Be warned that file equations can be a dangerous thing if you're not aware that you're using them. The important thing is not to forget when you have equations active. Using as few file equations as possible will help you keep track of what is being modified and keep your test run closer to the actual program processing conditions.

Another way to create a test environment is to set up an overall testing job to parallel your overall processing job. You can use the same menus for both "umbrella" jobs and have the same options available, but you can tailor the testing job to fit the testing environment, provided the environment is consistent for all of your testing.

For example, say part of your production job looked like this:

```
::
: RUN ?Program name?.EXE.PROD
:: USE ?Prompt answers file?.INCLUDE.PROD
::
```

then the corresponding part of your test job could look like:

```
::
: RUN ?Program name?.TEST.PROD
:: USE ?Prompt answers file?.TEST.PROD
::
```

At this point, you could build an overall umbrella job, with one of the first prompts being whether or not this was a test case. If testing, you could call the testing logic; if not, call the production logic.

Again, it is important that the test environment is as similar as possible to the production environment. If you create modular pieces of code, keeping things the same is easier to accomplish because you can use the actual production code in testing.

KSAM file, VPLUS file access

STREAMX can read KSAM files relatively easily using the VEFREADBYKEY function. You can take advantage of this for both KSAM files, and for VPLUS forms files, provided you first translate the VPLUS information into a KSAM file.

For VPLUS forms files, first translate the information into an ASCII flat file format. You can do this by using the Interex Contributed Software Library utility VPLUS2PC. This ASCII file can then be translated to a KSAM file to speed up access. Using the VEFREADBYKEY function, you can access KSAM information quickly and easily, and turn this information into usable STREAMX information.

Example KSAM read:

```
::
::ASSIGN KSAMFILE = "ACCTDEFX"
::ASSIGN FNUM = VEFOPEN ("{KSAMFILE},OLD")
::
::PROMPT STR KSAMKEY = "field to retrieve from KSAM"
::ASSIGN KSAMKEY = KSAMKEY + "                    "
::ASSIGN REC = VEFREADBYKEY({FNUM}, "{KSAMKEY[0:14]}",  0)
::ASSIGN VALUE = RTRIM(REC[15:65])
::ASSIGN VALUE = RTRIM(VALUE)
::
::ASSIGN VEFCLOSE({FNUM})
::
```

The above method works well in situations where you have to look up information in a table based on unique search keys. Make sure you re-create the KSAM file at the beginning of each run of your program to insure that you have the most current copy of the KSAM file information.

Conclusion

STREAMX can be a valuable tool in job creation and execution, as well as serve as a secondary programming language. Few tasks are beyond the capabilities of STREAMX, although STREAMX may not be the best tool in all cases. As with any set of tools, you must find the right tool for the job, and use it to its full potential.

STREAMX is also a great "structured language" for non-programmers, who can use programming concepts without all the overhead of a structured language. Non-programmers in our shop have been able to write their own STREAMX programs, saving the programming department extra work, and giving users a tool of their own.

As with any program or system being developed though, user involvement and input are crucial to the success or failure of the system's acceptance and usefulness. Keep users "in the loop" as far as design decisions go. Ask them to evaluate the feel of the new system, and

simulate the questions that they must answer when using the system. If the users don't like the product, they won't use it. No matter how sophisticated the system is, it's worthless if no one uses it.

As systems designers and implementers, avoid being constrained by the stated bounds of a product. Always try to find new uses for your tools, and you'll be more prepared than ever to face any challenge thrown at you. Don't fall for that "I y'am what I y'am and that's all what I y'am" stuff; that only applies in the cartoon world!

Advanced Program Examples

Appendices A and B are examples of some of the advanced programs used at MEDSTAT Systems. These programs show the concepts described in this paper, and also the degree to which the concepts can be applied. Only a portion of the actual code is included; full code for the following programs could (and does) fill a small book.

Appendix A -- DOITALL

This program is designed to run many different types of jobs, including custom ::USE files for the cases that don't fall under any other option. DOITALL is highly modular, as each box on the accompanying flow chart is a separate job. While source code is included for only some parts of the process, the concepts are the same for the whole program.

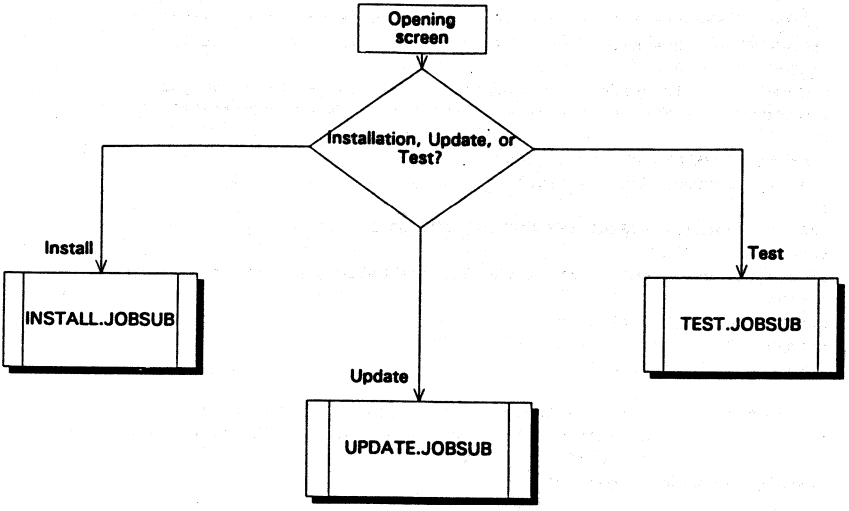
Appendix B -- DOITALL2

This program is an example of a table-driven program, with an example table to show the run. The table itself is documentation of the process; therefore, we can see exactly what was done during execution. This appendix also includes an example of a job-driving table.

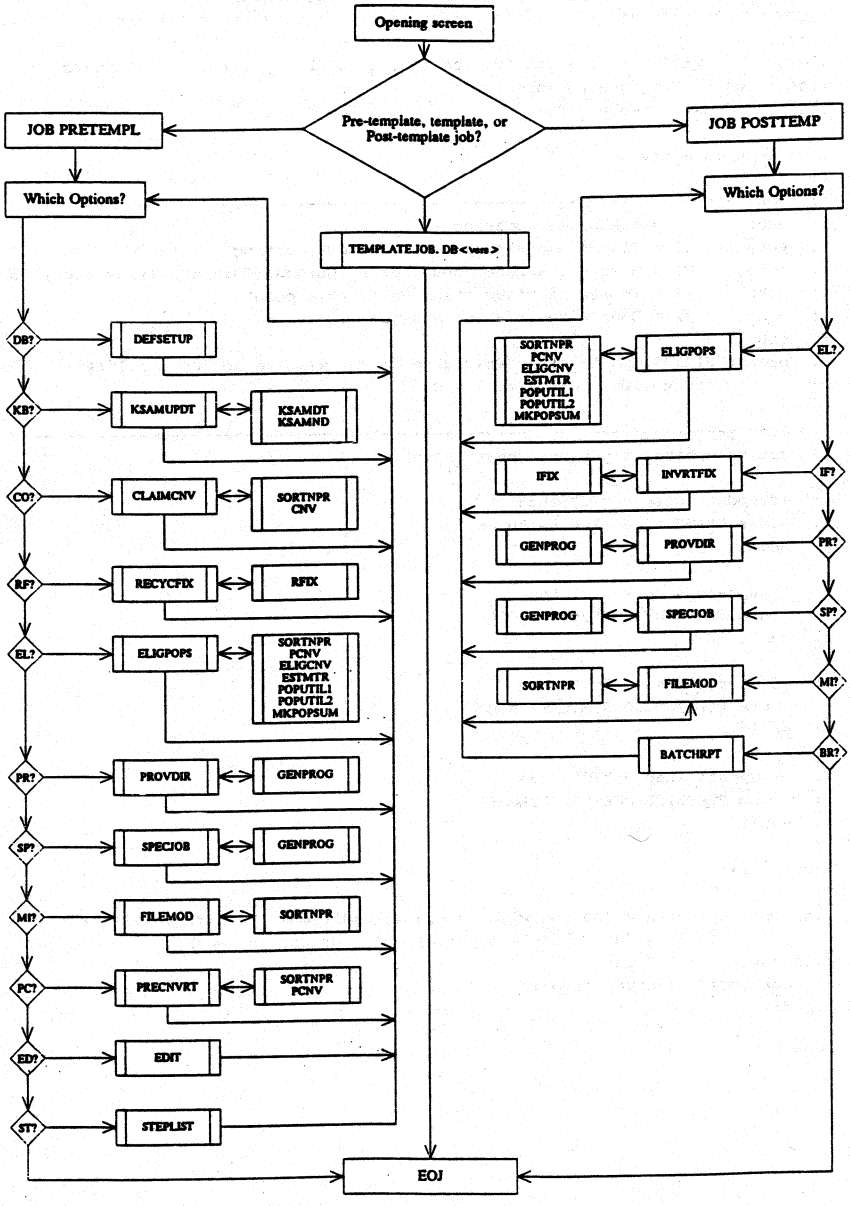
Appendix A -- DOITALL

```
::COMMENT ************************************************************
::COMMENT *    System:    The MEDSTAT Systems, Inc., Copyright 1992
::COMMENT *    Job:       DOITALL
::COMMENT *    Function: Driver program for database creation jobs.
::COMMENT ************************************************************
::
::setjcw STREAMXVARAREASIZE=11000
::assign current_dbver = "165"
::
:JOB {jobname},{HPUSER}.{HPACCOUNT};OUTCLASS = ,1
::echo
::echo Please select which of the options below you wish to perform:
::echo
::echo I = New Installation
::echo U = Database Update
::echo T = CP Program Testing
::
::prompt string jobtype = "Which type of job (I, U, T) [U]"; &
::               default = "U";check = (ups(jobtype)="I" or ups(jobtype)="U" &
::               or ups(jobtype)="T")
::assign jobtype = ups(jobtype)
::
::echo
::prompt string dbver = &
::       "DB version to use (DB version number only) [{current_dbver}]";&
::       default = "{current_dbver}"; check = (len(dbver) = 3)
::assign dbver = "DB" + dbver
::echo
::echo Using DBversion {dbver}
::echo
::
::echo ------------------------------------------------------------------------
::comment Determine which ::USE file to access
::echo
::if jobtype = "I" then
::  use install.JOBSUB.database
::endif
::if jobtype = "U" then
::  use update.JOBSUB.database
::endif
::if jobtype = "T" then
::  use test.JOBSUB.database
::endif
::
::comment Remove files no longer needed
::if fexists("acctdefx") then
::  purge acctdefx
::endif
```

DOITALL.JOB.DATABASE



Full UPDATE job diagram
(only a few options are detailed in the included code examples)



```
::COMMENT ************************************************************
::COMMENT *   System:   The MEDSTAT Systems, Inc., Copyright 1992
::COMMENT *   Module:   UPDATE
::COMMENT *   Function: Set up job for updating databases
::COMMENT ************************************************************
::echo
::echo This will create a job for updating a database, using the standard
::echo Custom Programming jobs.
::assign update_step = " "
::
::while update_step <> "//"
::  echo
::  echo ---------------------------------------------------------------------
::  echo     KS = KSAM file updating
::  echo     CO = Claims convert (with or without sorting)
::  echo     MI = Misc. file functions (Sorts, purges, Cust USE files etc.)
::  echo     ST = To see previous steps up to this point
::  echo     // = Done with Pre-template job
::  echo
::  prompt STRING update_step="What step do you wish to do (CO etc.)";&
::         check=(len(update_step) = 2)
::
::  echo ---------------------------------------------------------------------
::  assign update_step = ups(update_step)
::
::  if update_step = "KS" then
::    use KSAMUPDT.JOBSUB.DATABASE
::  endif
::
::  if update_step = "CO" then
::    use CLAIMCNV.JOBSUB.DATABASE
::  endif
::
::  if update_step = "MI" then
::    use FILEMOD.JOBSUB.DATABASE
::  endif
::
::  if update_step = "ST" then
::    use STEPLIST.JOBSUB.DATABASE
::  endif
::
::endwhile
::
::prompt string yn = "do you wish to run an additional job (Y/N) [N]"; &
::      default = "N"; check = (ups(yn) = "Y" or ups(yn) = "N")
::if ups(yn) = "Y" then
::  use addjob.jobsub.database
::endif
::
:EOJ
::
```

```
::COMMENT ************************************************************
::COMMENT *   System:   The MEDSTAT Systems, Inc., Copyright 1992
::COMMENT *   Module:   FILEMOD
::COMMENT *   Function: Modify files (sorts, copy's, purges, etc.)
::COMMENT ************************************************************
::
::echo   ------------------------------------------------------
::echo   -------------- SPECIAL/MISCELLANEOUS WORK -------------
::echo   ------------------------------------------------------
!comment ------------------------------------------------------
!comment -------------- SPECIAL/MISCELLANEOUS WORK -------------
!comment ------------------------------------------------------
::assign filemod_step = " "
::echo
::echo Ways to modify files:
::echo
::echo     SO = Sort a file
::echo     MF = Merge files
::echo     US = Custom ::USE file
::echo
::prompt string filemod_step = "Step to be done (SO, MF, etc.)"; &
::       check = (len(filemod_step) = 2)
::assign filemod_step = ups(filemod_step)
::
::if filemod_step = "SO" then
!comment ---------------- SORT FILE -----------------------------
::echo   ---------------- SORT FILE -----------------------------
::  use sort.jobsub.database
::  assign STEP_INFILE = sort_rawfile
::  assign STEP_OUTFILE = sort_sortfile
::endif
::
::if filemod_step = "MF" then
!comment ---------------- MERGE FILES ---------------------------
::echo   ---------------- MERGE FILES ---------------------------
::  prompt string STEP_INFILE = "(MULTIPLE FILES)"
:   ru SORTMATE
      I ?Files to merge (separated by commas)?
::  prompt string STEP_OUTFILE = "Merged output file name"
      O {STEP_OUTFILE}
      K 1,1
      E
::endif
::
::if filemod_step = "US" then
!comment ---------------- CUSTOM ::USE FILE ---------------------
::echo   ---------------- CUSTOM ::USE FILE ---------------------
::  use ?Full name of file to ::USE?
::endif
::
```

Appendix B -- DOITALL2

```
::COMMENT ************************************************************
::COMMENT *   System:   The MEDSTAT Systems, Inc., Copyright 1992
::COMMENT *   Module:   DOITALL2
::COMMENT *   Function: Driver program for database creation jobs
::COMMENT ************************************************************
::setjcw STREAMXVARAREASIZE = 11000
::prompt str tablename = "Job table file name "
::
::{(*****************************************************************)}
::{(*           List the job steps                                   *)}
::{(*****************************************************************)}
::assign fnum = vefopen ("{tablename},old")
::assign eof  = vefinfo ("{tablename}").eof
::
::assign step = 1
::assign i    = 1
::
::assign BUFFER  = vefread({fnum})
::assign title   = BUFFER[0:30]
::assign jobname = BUFFER[30:8]
::assign opname  = BUFFER[40:40]
::
:JOB {jobname},{HPUSER}.{HPACCOUNT},{dgrp};OUTCLASS = ,1
::echo
:
::while i <= eof
::  assign BUFFER    = vefread({fnum})
::  if BUFFER[0:1] <> "*" then
::    assign stepname    = BUFFER[0:20]
::    assign progname    = BUFFER[20:10]
::    assign step_infile = BUFFER[30:10]
::    assign COMMENT     = BUFFER[40:30]
::  endif
::  assign i= i + 1
::endwhile
::
::assign err = vefclose({fnum})
::
::{(*****************************************************************)}
::{(*            Set up the optional test sequencing                *)}
::{(*****************************************************************)}
::
::prompt str YN = "Is this a test run (Y/N) [N]";default = "N";&
::       check = (ups(YN) = "Y" or ups(YN) = "N")
::if ups(YN) = "Y" then
::  use ?File equations file name?
::endif
::
```

```
::{(*****************************************************************)}
::{(*           Set up the jobs according to the job table.          *)}
::{(*****************************************************************)}
::
::assign fnum = vefopen("{tablename},old")
::assign eof  = vefinfo("{tablename}").eof
::
::assign BUFFER = vefread({fnum})
::assign i      = 2
::
::while i <= eof
::  assign BUFFER    = vefread({fnum})
::  if BUFFER[0:1] <> "*" then
::    assign stepname    = BUFFER[0:20]
::    assign progname    = BUFFER[20:10]
::    assign step_infile = BUFFER[30:10]
::    assign COMMENT     = BUFFER[40:30]
::
:comment ************************************************************
:comment {stepname} - {COMMENT}
:comment ************************************************************
:comment
::echo
::echo ************************************************************
::echo  Set up for {stepname} - {COMMENT}
::echo ************************************************************
::echo
::
::assign stepname = ups(stepname)
::assign stepname = rtrim(stepname)
::assign stepname = ltrim(stepname)
::
::if stepname = "SORT" then
:   RU SORTMATE
      I {step_infile}
      O ?Name of sorted data?
      K ?Sort key?
      E
::endif
::
::if stepname = "CONVERT" then
:   RU {progname}
::  assign inclfile = progname[0:5] + "I" + progname[6:2] + ".INCLUDE.DATABASE"
::    echo inclfile = {inclfile}
::    use {inclfile}
::endif
:endif
::  assign i = i + 1
::endwhile
::
::assign err = vefclose({fnum})
:EOJ
```

Process table for DOITALL2

```
Client ABC      1Q92 UPDATE   CONVTEMP  Keith Raisch
*
SORT                          RAWDATA2  claims data sort
CONVERT             C1791O7A  SORTDAT   convert sorted data
CONVERT             C1792O7A  RAWDATA3  convert form II raw data
CONVERT             C1793OXA  RAWDATA4  convert form III raw data
*                                       (creates EDITIII, ZPRMIII)
```

Paper#....: **5040**
Title.....: **"WHAT'S IN A NAME ANYHOW ?"**
Author....: **GORDON W. GAVIN**
Company...: **JOHNSON HILL PRESS**
1233 Janesville Avenue
Fort Atkinson, WI 53538
Tel.....: **414-563-1736**
Fax.....: **414-563-1704**

The purpose of this paper is to present a common sense approach to managing an MIS Department. The target audience might be ...

- MIS Managers with little experience
- Developers of 'in-house' software
- Recently appointed MIS Managers who have inherited a shop with no formal naming conventions
- Those interested in a parameter driven Job Streamer

I make no excuse for the distinct lack of those impressive-sounding acronyms which proliferate the world of computing. And as for the content, well it is about as far removed from icons and graphical user interfaces as wet and windy Scotland is from hot and humid Florida. The material presented herein is a humble attempt to help managers of poorly organised shops establish some sort of order without having to spend an arm and a leg or whatever other parts of the anatomy might be marketable.

In 1978 I assumed my first data processing manager's position. The company had 'aged' custom-developed software with sparse documentation. There was no packaged software suited to the business and I was forced to embark upon writing all software in-house. Early in 1979, I derived a set of house-keeping practices which have served and continue to serve me and my employers for longer than I care to remember.

The nucleus is composed of SEVEN files, maintained within the confines of the systems department. These files are as follows:

1. **USER** - User Profiles which reflect the User's Identity (same as HPUSER), Name, Password, Telephone Extension, Log-on Menu. The idea is to dispense with the 'colon' prompt and always display a log-on screen which invites the user to enter a PASSWORD, prior to

displaying a Menu. By maintaining such a file, Passwords can be dynamically changed every month or whatever.

Please refer to a sample layout for such a file. It's purpose will perhaps become more apparent later. Note the use of fields which track the Date Created and the Date, Time and User Id for the last time the record was updated. These fields should be accommodated on every record on every file in a database.

```
File-USER-------"User-Profiles"

     Item UR_USER
          Begin Structure
               Item UR_INITIALS
               Item UR_DEPT_UNIT
          End Structure
     Item ZZ_FIRST_NAME
     Item ZZ_MID_INITIALS
     Item ZZ_LAST_NAME
     Item ZZ_TEL_EXTENSION
     Item UR_SECURITY_LEVL
     Item UR_PASSWORD
     Item UR_MENU_SIGN_ON
     Item UR_PORT_ID
     Item ZZ_DATE_CREATED
     Item ZZ_LAST_UPDATED
     Item ZZ_TIME_UPDATED
```

2. **MENUNAME** - Menu Names and Descriptions. Each User Menu is assigned a unique 4-character identification, preferably mnemonic. For example, Menu IMRR might be Inventory Management Report Request, Menu IMFM might be Inventory Management File Maintenance and so on. Similar functions are grouped together on the same menu. It is also preferable, but not always practically feasible, to limit the number of menu options so improving visual appeal and readability.

```
File-MENUNAME-------"Menu-Names"

     Item MX_MENU_NAME
     Item MX_MENU_DESC
     Item UR_USER
     Item ZZ_DATE_CREATED
     Item ZZ_LAST_UPDATED
     Item ZZ_TIME_UPDATED
```

3. **MENUOPT** - Menu Options. The description for each option reflects what is displayed on the menu. Narrative fields facilitate further explanation of the Menu Option. The 'key' to this file is 6-characters long and is the Menu Name suffixed by a 2-character option. For example, IMFM03 would be Menu IMFM, Option 03.

```
┌─File─MENUOPT─────"Menu─Options"─────┐
│                                     │
│      Item MZ_KEY                    │
│           Begin Structure           │
│                Item MX_MENU_NAME    │
│                Item MZ_MENU_OPTION  │
│           End Structure             │
│      Item MZ_MENU_OPT_DESC          │
│      Item UR_USER                   │
│      Item ZZ_DATE_CREATED           │
│      Item ZZ_LAST_UPDATED           │
│      Item ZZ_TIME_UPDATED           │
└─────────────────────────────────────┘
```

4. **MENUPARA** - Menu Option Parameters. A Menu Option may have execution-time parameters so that when a User chooses a particular Menu Option then he/she is prompted to respond to one or more questions. For example, for an Inventory Transaction Report a User may be asked to dictate a RANGE OF DATES for extracting Transactions within a particular time period. Each parameter record will have a prompt (i.e. the question being asked), a validity range for the response and a default response. The file index is an 8-character key which is a concatenation of the Menu Option with a 2-digit Parameter Number. This facilitates having upto 99 parameters per Menu Option. In fact I reserve Parameters 90-99 for global purposes such as User Id, Printer Id, # Copies to print, Type of Stationery, #Labels across a page, Job and Printer Priority and so on.

[Note if the Menu Option is submitted for Batch Processing then these parameters will be dynamically substituted into the Job Control Language to be streamed - see later.]

```
File-MENUPARA------"Menu-Option-Parameters"

      Item MY_KEY
           Begin Structure
                 Item MX_MENU_NAME
                 Item MZ_MENU_OPTION
                 Item MY_PARAMETER_NO
           End Structure
      Item MY_PARA_PROMPT
      Item MY_PARA_ALPHA_Q
      Item MY_PARA_LENGTH
      Item MY_PARA_FROM_VAL
      Item MY_PARA_TO_VAL
      Item MY_PARA_RANGE
      Item MY_PARA_DEFAULT
      Item UR_USER
      Item ZZ_DATE_CREATED
      Item ZZ_LAST_UPDATED
      Item ZZ_TIME_UPDATED
```

5. **PROGNAME** - Program Names. Once a Menu Option is selected, Programs are executed. For example, a Program may extract, sort and report Inventory Transactions. User parameters may dictate which Part Numbers are to be extracted, how to sort the data and how many copies to print. Program Names are 8-characters long, a concatenation of the Menu Option with a 2-digit Program Number. For example, Programs IMRR0401 & IMRR0402 (group OBJECT) relate to Menu IMRR, Option 04 where Program 01 may extract the data and Program 02 generate the report. Note that the source programs would be stored with the same names in group SOURCE.

[Some Menu Options may facilitate submitting a JOB for batch processing. The file to be streamed would have an 8-character name which would be similar to actual Program Names, viz. a concatenation of the Menu Option with a 2-digit 'Job Control Language' File Number, for example file IMRR0401 (group JCL) may be streamed from Menu IMRR, Option 04 and so on. Note that run-time parameters (courtesy file MENUPARA) would be prompted for and 'substituted' in the file before streaming. This suggests adopting a standard for embedding parameters in a Job Stream. I use 'curly braces' so that {01} would be parameter 01, {02} would be parameter 02 and so on. The program scans each command line to be streamed, searching for the opening brace '{' and if found, dynamically substitutes the User's response to that parameter prompt. And so on until each

line has been fully 'parsed'.]

```
File-PROGNAME-----"Program-Names"

     Item PX_KEY
          Begin Structure
               Item MX_MENU_NAME
               Item MZ_MENU_OPTION
               Item PX_PROG_NUMBER
          End Structure
     Item FI_FILE_NAME
     Item PX_PROG_DESC
     Item UR_USER
     Item ZZ_DATE_CREATED
     Item ZZ_LAST_UPDATED
     Item ZZ_TIME_UPDATED
```

6. **FILENAME** - Data File Names. A record is created for each permanent data file. This same record contains a description of the file and perhaps even the Maximum Number of Records for allocation. By maintaining this file, a relationship between PROGRAM and FILE Names is established. It becomes very easy to gauge the impact of having to amend a record layout by simply reviewing all of the affected Programs.

[Such a facility promotes systems integrity and is particularly useful in a changing environment, a sort of quasi documentation.]

```
File-FILENAME-----"File-Names"

     Item FI_FILE_NAME
     Item FI_FILE_DESC
     Item FI_FIELD_PREFIX
     Item FI_FILE_REC_CAP
     Item FI_FILE_REC_USED
     Item UR_USER
     Item ZZ_DATE_CREATED
     Item ZZ_LAST_UPDATED
     Item ZZ_TIME_UPDATED
```

7. **JOBS** - Job Numbers. Each time a User elects to submit a Job for BATCH PROCESSING, a record is created with a key which is a concatenation of HP's Job Number and the Date. This file facilitates printing a 'JOB ORDER FORM' for the Computer Operator, detailing the User's response to Parameter prompts (MENUPARA) and any special instructions from the User. This file is also used in combination with HP's log files to facilitate internal billing for computer time and resources. See later.

```
File JOBS "Batch Jobs"

      Item JB_KEY
           Begin Structure
                Item JB_JOB_NUMBER
                Item ZZ_DATE_CREATED
           End Structure
      Item MZ_KEY
           Begin Structure
                Item MX_MENU_NAME
                Item MZ_MENU_OPTION
           End Structure
      Item UR_DEPT_UNIT
      Item GL_ACCOUNT_NO
      Item JB_DATE_REQUIRED
      Item JB_TIME_REQUIRED
      Item JB_PRINT_COPIES
      Item JB_LABEL_TYPE
      Item JB_LABEL_ACROSS
      Item JB_MAG_TAPE_TYPE
      Item JB_DISKETTE_TYPE
      Item JB_INSTRUCTIONS
      Item JB_LINES_PRINTED
      Item JB_TRANS_ADD
      Item JB_TRANS_DEL
      Item JB_TRANS_MOD
      Item UR_USER
      Item ZZ_LAST_UPDATED
      Item ZZ_TIME_UPDATED
```

Perhaps we should take a look at how the aforementioned melds together to help organise a computing environment. Users may be identified by their Initials suffixed by their Department Number, for example JD510 might be John Doe in Department 510. John might log on thus ... 'HELLO JD510.MIS' and the Account UDC would respond with the log on screen, inviting the user to enter a PASSWORD and perhaps a MENU if the User is

qualified to access more than one menu, otherwise the default menu is displayed. See the log-on Screen, referenced 'SYM0' in top right hand corner. The User is given three attempts at the password, thereafter the log-on simply aborts and a message is sent to the operator.

MODE: ACTION: Ref. SYM0

```
                    Johnson Hill Press
                      MIS Department



      User Id.........:  GG630     Gordon W Gavin
      Password........:
   01 Publication.....:  244       PRO Magazine
   02 Menu Name.......:  adm1      AD SALES
```

Assuming the User enters a valid password, a Menu is displayed (see Screen referenced 'ADM1'). Menu and Program Screens are laid out in a standard and consistent fashion. In the example shown, the Date and Time appear on the top left hand corner. The Menu Name, the User Id and the Port# (workstation) appear on the top right corner. The Menu is entitled 'AD SALES' and guess what, this title is retrieved from file MENUNAME! Similar Menu Options have been grouped together under four sub-headings, viz. 'Order Entry', 'File Maintenance', 'Issue Functions' and 'Other'. The first two categories deal with on-line transaction entry. If a User elects to enter Menu Option 03, 'MISCELLANEOUS ORDERS', then the first PROGRAM to execute will be ADM10301. The Program Name is a concatenation of the Menu Name, the Menu Option and a Program Number ... 'ADM1' + '03' + '01'. If other programs are invoked to service this same Menu Option, then the Program Names simply run in sequence - ADM10302, ADM10303, and so on.

MODE: ACTION:19 Ref. **ADM1**

Date 06/25/93	PRO Magazine	User GG630
Time 8:07:53	AD SALES	Wksn 214

Order Entry	Issue Functions
01 CONTRACTS	13 ACKNOWLEDGEMENTS
02 SPACE ORDERS	14 ISSUE PLACEMENTS
03 MISCELLANEOUS ORDERS	15 SALES BOOKED
04 COMPETITOR ADS	16 AD SIZE ANALYSIS
	18 PRELIM BILLING
File Maintenance	**19 ADVERTISER INDEX**
	20 CLOSE ISSUE
05 AD CAPTIONS,PRODUCTS	
06 AGENCIES	Other
07 CLIENTS	
08 CLIENT/AGENCY PRODUCTS	95 Display - QUEUES
09 CONTACT NAMES	96 Menu ADM3- MANAGER
10 PRINT-TEXT MESSAGES	97 Menu ADM4- FEATURES
11 EDITORIAL FEATURES	98 Menu SAM1- ANALYSIS
12 SALES DIARY	99 DATABASE INQUIRY

The Menu sub-heading 'File Maintenance' simply facilitates Adding, Deleting or Modifying records interactively.

'Issue Functions' suggest actions which perform specific tasks, tasks which probably will generate printed output. Menu Options 13 through 20 may be run either in BATCH or SESSION.

The last Menu category is 'Other' which offers miscellaneous options such as displaying Job and Print Queues or calling other menus in order to navigate across a system.

To illustrate the use of files MENUOPT and MENUPARA, let's select Menu Option 19, the 'ADVERTISER INDEX'. The User is presented with a list of questions, courtesy of Screen SYM00000 (refer). The top half of this screen confirms the Menu, the Option chosen and a Description from File MENUOPT. The questions in the upper portion of the screen reflect a common set of questions which invite the User to indicate 'Batch or Session', 'Job & Printer Priority', 'Other Instructions' and so on. The questions in the lower part of the display are extracted directly from File MENUPARA ... these constitute the run-time parameters.

Note that the default response is displayed to help the User.

MODE: ACTION: Ref. SYM00000

Menu: ADM1	Opt: 19	ADVERTISER INDEX

BatchJob/Session..:	B	Job Priority...:	8
Date Required YMD.:	930625	Time Required..:	9:30
Preferred Printer.:	251	#Copies.....:	1
Label/Forms Type..:	BLUEBAR	#Labels Across.:	
Printer Priority..:	8	Mag Tape reqd..:	N
Other Instructions:	Sample for INTEREX '93		

Question	Answer	Default
01 Publication ISSUE# ?	9306	
02 Print NAME only ?	N	Y
03 Create TYPE-SET file ?	Y	N

In the example shown, the User elected to submit a BATCH JOB. The parameters are dynamically substituted into file ADM11901.JCL.MIS which is then streamed for processing.

The Job Control Language file looks something like this, PRIOR to parameter substitution -

```
!JOB ADM11901,{98}.MIS,AD;INPRI={92}
!Comment
!Comment Job Name: ADM11901.JCL
!Comment Desc....: This job will generate an
!Comment           ADVERTISER INDEX for an Issue
!Comment Params..: 01= {01} - Issue to select
!Comment           02= {02} - Co. Names ONLY ?
!Comment           03= {03} - Create TYPESET ?
!Comment           98= {98} - User Identity
!Comment           99= {99} - Publication Code
!Comment Files...: ADVERT   - Ad Insertion Orders
!Comment           ADVI{99} - INDEX to TYPE-SET
!Comment Programs: ADM11901 - extract Clients
!Comment           ADM11902 - print INDEX and
!Comment                      create TYPE-SET
!Comment Reports.: Ref ADM11902- Advertiser Index
!Comment Progrmmr: Gordon Gavin- 29th Nov. 1989
!Comment Rev #1..:
!Comment---
!Comment   Select Advertisers for the Issue
!Comment
!Setvar  QTYPESET '{03}'
!If      QTYPESET = 'Y'
!     Build  ADVI{99};REC=-64,32,F,ASCII;DISC=512
!Endif
!File   PUBISSUE;SHR;LOCK
!File   ADVERT;SHR;LOCK
!File   CLIENT;SHR;LOCK
!File   ADM119;DEV={97},{93},{96}
!Run    ADM11901.OBJECT.SYSTEMS
.................................
.................................
!Eoj
```

and AFTER parameter substitution, like this -

```
!JOB ADM11901,GG630.MIS,AD;INPRI=8
!Comment
!Comment Job Name: ADM11901.JCL
!Comment Desc....: This job will generate an
!Comment           ADVERTISER INDEX for an Issue
!Comment Params..: 01= 9306 - Issue to select
!Comment           02= N - Co. Names ONLY ?
!Comment           03= Y - Create TYPESET ?
!Comment           98= GG630 - User Identity
!Comment           99= 244 - Publication Code
!Comment Files...: ADVERT   - Ad Insertion Orders
!Comment           ADVI244 - INDEX to TYPE-SET
!Comment Programs: ADM11901 - extract Clients
!Comment           ADM11902 - print INDEX and
!Comment                      create TYPE-SET
!Comment Reports.: Ref ADM11902- Advertiser Index
!Comment Progrmmr: Gordon Gavin- 29th Nov. 1989
!Comment Rev #1..:
!Comment---
!Comment   Select Advertisers for the Issue
!Comment
!Setvar  QTYPESET 'Y'
!If      QTYPESET = 'Y'
!     Build  ADVI244;REC=-64,32,F,ASCII;DISC=512
!Endif
!File   PUBISSUE;SHR;LOCK
!File   ADVERT;SHR;LOCK
!File   CLIENT;SHR;LOCK
!File   ADM119;DEV=251,8,1
!Run    ADM11901.OBJECT.SYSTEMS
.................................
.................................
!Eoj
```

By reviewing the before and after, you will see where dynamic parameter substitution has occurred.

Meanwhile back at the ranch (computer room), a COMPUTER JOB ORDER has been printed for the Operator's convenience and File JOBS has been accordingly updated.

Refer to the sample COMPUTER JOB ORDER form which records the User's instructions and parameter responses. This form can be married to whatever other materials are to be despatched back to the user (refer SYM00002.)

```
Date 06/25/93    COMPUTER JOB ORDER      Ref. SYM00002
Time 08:08:42    Johnson Hill Press      User GG630
-------------------------------------------------------

Job Number.....:  0822     [ Menu: ADM1  Option: 19 ]

Job Description:  ADVERTISER INDEX
                  for a particular Publication Issue

Department.....:  244      PRO Magazine

User Name......:  Gordon Gavin    [ Job Priority: 8 ]

-------------------------------------------------------

Date Required..:  93/06/25

Time Required..:   9:30

-------------------------------------------------------

Printer Device#:  251           [ Print Priority: 8 ]

#Copies........:    1

Labels / Forms.:  BLUEBAR

-------------------------------------------------------

Mag Tape reqd ?:  N

-------------------------------------------------------

Other Instructn:  Sample for INTEREX '93

-------------------------------------------------------

Parameter # 01 :  Publication ISSUE# ? ......  9306
            02 :  Print NAME only ? .........  N
            03 :  Create TYPE-SET file ? ....  Y
```

Finally, let's have a look at the report generated by this Job. As you may recall, screens adopted a standard format and reports are no exception. See sample report Ref ADM11902, created by Menu ADM1, Option 19, Program 02. The Date and Time on the left, the Menu Option description is the Heading and the User Id and Program Name on the right alongside the Page Number. Reports should always have an '- END REPORT -' message clearly displayed.

Date 06/25/93 **ADVERTISER INDEX** User GG630 Page 5
Time 09:13:21 for Ref. ADM11902
244 - **PRO Magazine**
Issue# 9306 - **June/July 1993**

Advertiser Name	Page#
TECUMSEH PRODUCTS CO. 900 North Street Grafton, WI 53024-0724 Tel 414-377-2700 Fax 414-377-4485	23
TENNESSEE NURSERYMEN'S ASSOC. P O Box 57 McMinnville, TN 37110-0057 Tel 615-473-3951 Fax 615-473-5883	41
WALKER MFG. CO. 5925 East Harmony Road Fort Collins, CO 80525-1362 Tel 303-221-5614 Fax 303-221-5619	17

Total #Advertisers: **46** Total #Ads: **49**

- E N D R E P O R T -

Execution-time parameters provide tremendous flexibility to Users and can contribute to reduced paper usage, processing times and programming requests. Parameters can be as simple or as sophisticated as the application dictates.

Just to give you a flavour for a more powerful set of parameters, I have included a partial extract from a particular Menu ADM4, Option 01. Refer. This Option offers the User the ability to print a Report and/or Mailing Labels, generate a Magnetic Tape for an Ink-Jet Machine, Download a File to a PC in a choice of formats and to Sort the data in a particular order ... and that's only the first 10 parameters !

MODE: ACTION: Ref. SYM00000

Menu: ADM4	Opt: 01	COMPANY CONTACTS

BatchJob/Session..:	B	Job Priority...:	12
Date Required YMD.:	930601	Time Required..:	10:30
Preferred Printer.:	6	#Copies.....:	2
Label/Forms Type..:	ADHESIVE	#Labels Across.:	3-up
Printer Priority..:	10	Mag Tape reqd..:	N
Other Instructions:	Sample with MORE PARAMETERS		

Question	Answer	Default
01 Optional REFERENCE Id	SAMPLEX	
02 Print LIST OF NAMES chosen?	N	Y
03 Print MAILING LABELS ?	Y	Y
04 Print BAR CODE on Label ? .	N	N
05 Create INK-JET Tape ?	N	N
06 FILE NAME for PC DOWNLOAD ?	INTX	
07 PC FORMAT ? [1=WordP 2=Asc]	2	1
08 1st SORT ? [1=Slsrep 2=Comp	5	1
09 2nd SORT ? 3=Name 4=Zip	6	2
10 3rd SORT ? 5=State 6=Rank]	2	3

Two other subjects which merit mention are formalising User requests for Systems and Programming Services and also maintaining a register of PC Hardware and Software. The former can be achieved very easily by facilitating a menu Option to enter such requests into a file PROJECTS for example. The Project is assigned a UNIQUE Number such as the Date & Time and the User enters a description with some sort of benefits rationale and a required date. The computer prints a Request Acknowledgement for return to the User while TWO copies are despatched to the systems manager for scheduling and allocation to a programmer. One of these copies is given to the programmer to act as authorisation to commence work while the other is

retained by the manager in a projects 'pending' file. Programmers record their time (sometimes not a popular mandate) against each project. By so doing, it is a simple matter to generate a Project Status Report and also to bill back internally to the requesting user department for systems work. See printed Request sample, reference PJM10102.

Date 06/25/93 **SYSTEMS REQUEST FORM** User BG210
Time 11:56:02 Johnson Hill Press Ref. PJM10102

Project Number.: **930625.1155**

Department.....: **210 Equipment Today**

Project Type...: **NEW PROGRAM - ONE TIME**

System Module..: **SA - Sales Analysis**

Summary Desc...: **This is merely a sample**

Benefits.....: **Help establish criteria for deriving a new Rate Card.**

Detailed Desc..: **Please provide a list of Advertisers who have run 6 or more Ads in Competitor Pubs over the last 5 years.**

Date Required..: **93/06/30**

Latest Date....: **93/07/07**

Requested By...: **Bonnie Gable**

On Behalf of...: **Tom Swetland**

Assigned Prgmr.:

Estimated Hours:

Planned Start..:

Much as I respect Personal Computers, they have at times been a 'thorn in my side'. Over the last three or four years, my PC-base of Users has grown dramatically, so much so that I was fast losing track of monitors, modems, laptops, software, etc.. The fixed asset system within the accounting department just could not capture the information I required such as additional diskette drives, warranty information, repair records and so on. Again it was not difficult to establish a couple of files, MISEQUIP and MISMAINT, to hold such information. A combination of Serial and Model Numbers facilitate a unique key for each piece of equipment and, in combination with file USER, a policing system was developed to keep a handle on who had what equipment. This system also helps combat software piracy. Incidentally, if your company's employee handbook does not contain a paragraph on this very topic then I'd recommend that you remedy this situation.

The material detailed herein is hardly earth-shattering, in fact I would be first to admit that it's rather olde worlde. It is a simple, straight-forward approach to computing which does work and costs little to implement. The last figure - Ref. SYM8 - is a Systems Menu which facilitates maintaining the files discussed herein.

MODE: ACTION: Ref. SYM8

Date 06/25/93 Johnson Hill Press User GG630
Time 17:23:42 **SYSTEM CONTROL** Wksn 214

File Maintenance	Report Requests
01 USER PROFILES	21 USER PROFILES
02 MENU NAMES	22 MENU NAMES
03 MENU OPTIONS	23 MENU OPTIONS
04 MENU OPTION PARAMETERS	24 MENU OPTION PARAMETER
05 PROGRAM NAMES	25 PROGRAM NAMES
06 FILE NAMES	26 FILE NAMES
07 JOB NUMBERS	27 JOB NUMBERS
08 USER FIELD DEFINITIONS	28 USER FIELD DEFINITION
09 USER PUBLICATIONS	29 USER PUBLICATIONS
10	30
11 MIS EQUIPMENT REGISTER	31 MIS EQUIP REGISTER
12 MIS EQUIPMENT MAINT	32 MIS EQUIP MAINTENANCE
13	
14	98 Menu SYM0 - MODULES
15 REASSIGN MENU OPTIONS	99 DATABASE INQUIRY

Paper #5042

Secrets of MPE V Tables *III*: Square Pegs for Round Holes

Craig Nickerson
United Electric Controls Co.
180 Dexter Ave.
P.O. Box 9143
Watertown, MA 02272-9143
U.S.A.
(617) 926-1000

The greatest pleasure comes from doing what people say you cannot do.

- Found in a fortune cookie

Introduction

The original paper on MPE's internal data structures was written in 1985 by Eugene Volokh of VESOFT, Inc. and bore the title "Secrets of System Tables ... Revealed!"[1] Once having fathomed the intricacies of MPE V's system of managing code segments, and eager to share this knowledge, I wrote a sequel entitled "MORE Secrets of MPE V Tables: A Closer Look at Code Management Structures"[2] (hereinafter referred to as *S-II*) for the INTEREX '92 Conference in New Orleans. Only after final submission, however, did I realize that despite its length, my opus was less exhaustive of its subject than it could have been.

Having prepared a supplement to hand out at my presentation--which never took place, due in no small part to Hurricane Andrew--I decided to develop it into a separate paper, broadening its scope to include other areas where knowledge of structures normally transparent to the applications software engineer can help you get more yet from your Classic system. But the learning process never stops, one possibility paves the way for another, and the tale grows in the telling; and so the present work has blossomed into ten sections, not directly related to each other.

Most of these applications involve structures which are local to the current process or the program being run, and do not require any special capabilities other than Data Segment Management (DS). The few that require Privileged Mode (PM) capability must be actualized with extreme care and with all due regard for system security.

The information here presented is based on my experience with MPE V as far as version V-Delta-4. As this paper is conceived as the third in a series, your familiarity with the other two would be helpful.

Custom operations to access data in structures out of the

convenient reach of the third-generation languages (3GLs) we have at our disposal should be *encapsulated* in library procedures that are *reusable* in future application programs and systems. In the case of privileged operations to be performed on behalf of non-privileged programs and logons, very much in order is a discussion of--

"OPTION PRIVILEGED"

--Of what it is, how it works, and the risks inherent in its use as an encapsulation tool.

For security reasons, most application programs do not have PM capability; yet, without any privilege at all, it is impossible to even create and access your own files. For this reason, MPE provides access to privileged code by allowing specially-marked code segments to run in Privileged Mode unconditionally.

The "PRIVILEGED" keyword of the OPTION clause is included in SPL source code when procedure design and purpose require execution in Privileged Mode regardless of the run mode of the calling process, and if SL-resident, also regardless of the capabilities of the sharing program. FOPEN is one of many examples of commonly-used "OPTION PRIVILEGED" (O/P) procedures.

This option directs the compiler to set bit 2 of word 2 of the primary entry point's Directory entry in the USL. When, however, the code segment containing the procedure is *prepared*--either by PREPping the USL containing it, or by adding it to an SL--the quality of privilege becomes an attribute, not of the individual procedure, but of the entire code segment. This means that all other procedures within that segment, regardless of options, will run in Privileged Mode as though they were also "OPTION PRIVILEGED". It takes only a single unit to have this effect on its "housemates".

Installing a sharable O/P procedure requires that you be logged on with PM capability, AND that both the group and account where the SL resides have PM capability; but once installed, anyone may call it who has EXECUTE access to the SL and is in a position to LOADPROC the procedure, unless it is also "OPTION UNCALLABLE".

Acquiring privilege in this manner is a two-edged sword. Unlike that obtained by calling GETPRIVMODE, this kind of privilege is permanent, from the time you enter the code segment at any point until the time you exit from it. The automatic security- and bounds-checking mechanisms that protect non-privileged processes from themselves and each other, and the operating system from all of them, are on holiday for the duration of your visit! The same circumstance prevails in any other code segment called from a state of permanent privilege.

To reduce the need for manual bounds-checking in an O/P procedure, formal parameters that are strictly for input should

be constructed as *by value*. If there is a non-character datum to be returned to the caller other than a Condition Code, the procedure should be typed and the datum returned as the function value. A passed and/or returned reference parameter must be minimally bounds-checked to ensure that no part of it resides below DL or above Q-4, especially if the caller is running non-privileged; the caller's run mode is indicated by bit 0 of the old status word at Q-1.

An O/P procedure for general use should only be installed in the system library (SL.PUB.SYS); not only is it the one SL with PM capability that is available to all programs and logons, it is also the most secure against tampering and abuse.

On general security principles, I would advise that only those personnel whom you can trust as you would a System Manager have access to the PMAPs of privileged code segments. If you have similarly restricted 3GL programming, I would further advise securing EXECUTE access to the Segmenter (programs SEGDVR.PUB.SYS and SEGPROC.PUB.SYS) and the contributed program SLPMAP, as we have done in our shop.

I. Interrogating Program Capabilities

The WHO intrinsic tells you everything you would want to know about the capabilities possessed by your logon. Unfortunately, the presence or absence of a particular *general resource* capability at logon level gives you no advance warning as to whether a call to an intrinsic requiring that capability will abort your process, because the required capability is usually an attribute only of the program that you're running. In writing library procedures to be shared among a variety of programs, I've encountered numerous situations where constructive alternatives to aborting the process in the event of deficiency of program capability are highly desirable.

In User Mode, it is possible under the right conditions to open the program file and read the capability mask from record 0, word 0. The full name of the current program file can be obtained by calling the PROCINFO intrinsic, provided that your logon has READ access to it. FOPENing the program file[4] further requires LOCK access in order to avoid collision with concurrent :RUN commands or CREATE(PROCESS) calls by other users. And if the program file is, or may be, protected by a lockword, then that must somehow be provided for as well.

In Privileged Mode, PROCINFO provides its full range of services regardless of logon capabilities or file access limitations. The undocumented MUSTOPEN procedure may be called instead of FOPEN to bypass file security and any lockword that may be in effect.

But if you're running privileged, why settle for second-rate when first-rate is attainable? The program's capability mask is copied at process-creation time to your stack, and may be read by an O/P procedure from the Process Control Block eXtension (PCBX). The location of the mask in PXFIXED is clearly identified in the Tables manual, Chapter 7.

II. Dynamic NOCB: More Stack Space Without Impairing Overall Performance

One problem we've encountered in the course of customizing our ASK MANMAN[5] system is that every once in a while, a "Dispatcher"-resident sub-application would run out of stack space during a sort.

Many subroutine-type commands use a plethora of scratch files, especially for multi-level explosions of Bills of Materials, where-used lists for components and allocations for work orders; and in some cases, scratch files are kept open, their file numbers saved on global stack, when a command subroutine exits to the main program, to maximize response time when the command is re-entered. When these files are sorted, more scratch files are opened by the SORT/MERGE intrinsics, which also allocate temporary global storage at the top-of-stack. All of this file access, on top of opens maintained on databases and administra-

tive files, plus the humungous amount of global storage used by the program itself, mandate a stack of the maximum allowable size. Occasionally, even that isn't quite enough!

One fairly simple way to alleviate the problem is to run MANMAN with the NOCB ("NO Control Blocks on the stack") option. A name like this bears some explaining; I will try to make a long and complicated story as short as possible:

In Chapter 6, the Tables manual describes, with more than its usual degree of clarity, the data structures internal to the File System. It is in the PXFILE section of your PCBX that all of your file access is managed. PXFILE has two subsections that tend to get bigger as more files are opened:

* The Active (or Available) File Table (AFT); one 6-word entry for each file access path your process has generated, and indexed by MPE file number, negatively from the physical end of PXFILE. In the case of a KSAM file, the file number returned by FOPEN points to a KSAM-type entry pointing in turn to two regular entries, one each for the key and data file access paths. Also, if you've DBOPENed a database, you have an AFT entry for the root file.

* A Control Block Table (CBT); 8 words of overhead, plus an additional 8-word overhead for each control block that the File System has seen fit to put in here.

These are the control blocks that are usually placed in your stack-resident CBT:

* One 52-word Physical Access Control Block (PACB) for each access path opened for non-multi-access. A PACB may include additional space for optional buffering.

* One 16-word Logical Access Control Block (LACB) for each access path opened for MULTI or GMULTI access, including those opened on $STDIN and $STDLIST.

* One File Control Block (FCB) for each disc-resident file opened for exclusive access; most scratch files used by MANMAN sub-applications fall into this category. An FCB ranges in size from 38 to 100 words, depending upon extents allocated.

To make room for a new AFT entry or control block after the Available area is used up, PXFILE is expanded, pushing everything on the stack at and above DL into higher segment-relative locations. Consequently, an FOPEN or DBOPEN call is doomed to failure if the DB-relative address pointed to by the Z register (highest-ever top-of-stack) would be forced to a segment-relative offset beyond the physical end of the stack. A call to ZSIZE with a 0 parameter can help prevent this from happening, but only provided that Z is not already coincident with S; and if the real problem is not enough space between Q and the physical limit,

then calling ZSIZE is not the answer.

Now, what the NOCB option does for you, when specified in a :RUN command or CREATE(PROCESS) call, is to prevent control blocks from being formed on your stack, thus greatly reducing the need to expand PXFILE. Forcing them into other data segments, however, may give rise to a degradation of performance, mainly because of the increased incidence of the *absence trap*; this happens when a required data or code segment is currently swapped out and not in main memory.

We had two problems with running the MANMAN Dispatcher with NOCB: 1) We did not want to noticeably degrade response time by applying NOCB across-the-board, for both frequently- and infrequently-accessed files; 2) with very few exceptions, the Dispatcher is run from logon menus, all of which are already hard-coded to run it without NOCB.

So, how could we use this feature to make more room for the sorts? In looking about in the Tables manual for the NOCB indicator, I was immensely pleased to find that it is process-local—not part of the program's load options, which are fixed for every sharing process. It is a single bit in the PXFILE overhead area, and for each FOPEN call, its current state influences the placement of any new control block(s) required. So, all that's needed is an O/P procedure that can be called to set or clear this bit on a selective basis.

So far, this approach has been working well for us.

III. Another "Hook Under the Hood": Trapping Procedure Exits

As part of our general program of simplifying our company's design and manufacturing operations, we have undertaken to reduce the variety of our component parts and subassemblies. To help this process along, I have enhanced our MANMAN system to support lookups on part sets, as well as individual parts, without having to exit MANMAN to run an *ad hoc* QUIZ report.

Part sets, like file sets, involve the MPE wild cards "?", "#" and "@". To match and gather a set[7] as rapidly as possible from among more than 40,000 parts, a flat Parts List file is used instead of the Item Master set in the database. Our module for prompting for, and gathering, part sets is a procedure called GENITEM, which is designed as a superset of the MANMAN utility procedure used to prompt for, and validate, a part number; this to facilitate upgrading of existing command handlers which prompt for a part number at the top of a transaction loop.

When a command handler calls GENITEM, it is always, as with the indiginous module, to obtain an individual part number. If the design engineer has specified a part set, the part number returned is the next one that matches the set; this means that GENITEM has to maintain state information, including the Parts List and

part set file numbers, on global stack between calls. Now, it's easy enough for GENITEM to know when it's "clean-up" time if "E" (for "exit") is entered at the "PART NUMBER OR SET?" prompt, or if some serious error condition is detected within GENITEM; it is not so easy if the command handler sees fit to abandon processing in mid-loop, but "forgets" to let GENITEM know what's going on; after all, globally-stored state information cannot reset itself, and I also wanted GENITEM to rely as little as possible on caller coöperation. So, I looked about for a way to hook the command handler in the act of RETURNing to the command parser, to ensure that GENITEM is properly re-initialized the next time any sub-application that uses it is invoked.

For the third consecutive INTEREX, I find myself going on about that most accessible of all system tables, the *stack marker*. There is a way to make an EXIT instruction, RETURN statement or GOBACK verb do what it was never intended to do--call any procedure you want it to! The trick is to manipulate the stack marker's saved information about where process execution is to resume when a procedure exits.

MPE allows you to branch via the EXIT instruction to any valid PB-relative address in any loaded code segment shared by the current process; the only restriction is that you cannot in any way EXIT into Privileged Mode from User Mode. You are permitted to change the contents of any stack marker before it is EXITed through; you may even EXIT through any valid stack marker that you have built and linked yourself and to which you have forced the Q register to point. The thing to watch out for is that the old status word in a stack marker other than the most recent in the current trace-back chain can be written to in User Mode only if the mode of addressing is indexed, i.e. if it's referenced as an array element. These are important points to remember in later sections.

Our exit-trapping solution provides for multiple exit traps at different levels by means of an "exit trap stack", for saving original return vector information from hooked stack markers (segment number, mapping flag, return address), and maintained within a shared Utility Table structure (UTAB) residing either in a process-local extra data segment (XDS) or in a rather interesting stack-resident object called a "Green area"--more about this, and why it is so called, in Section IX. Location DB-14--carefully chosen to avoid collision with the various subsystems we have, like VPLUS and FORTRAN 77, that make extensive use of the "heap"--is used to hold either an XDS logical index or a special flag value, depending upon where the UTAB has been built. In MANMAN, which has already exhorbitant stack requirements, we use the XDS, but the processing overhead is not of grave concern here; the issue is what happens between the last transaction of one cycle and the first of the next.

The exit trap stack starts with a header entry whose word-by-word contents are:

0 - Current number of trap entries.
1 - Maximum number of trap entries (constant).
2 - Trap-in-progress flag.
3 - Entry size (4).

The header entry is always the same size as a trap entry, so that the table-relative offset of the most recent entry may be obtained by just multiplying the current entry count by the entry size (which has ended up in word 3 due to the evolutionary nature of the development process).

Each trap entry is layed out as follows:

0 - DB-relative address (or Q) of the trapped stack marker.
1 - Original return address and mapping flag saved from the stack marker.
2 - Original segment number saved from the stack marker.
3 - *Plabel* of the procedure to be called when this trap is taken.

Like the stack markers themselves, exit trap entries are created and deleted on a LIFO basis.

A procedure may trap the exit from its caller by calling library procedure EXITRAP, passing the *plabel* of a subroutine designed to do whatever clean-up operations need to be done. EXITRAP, in turn, calls another library procedure called BLDUTAB, which, as its name implies, creates and initializes the UTAB if none already exists; this is done in the Green area if it exists, is large enough and is not internally flagged as reserved, otherwise the XDS is used. EXITRAP then checks to make sure that the saved address in the current, if any, trap entry does not match the Q of the stack marker to be trapped; if everything's OK, the new trap entry is created and the entry count in word 0 of the header entry incremented by 1. The return vector in the stack marker, after being saved in the new trap entry, is then altered to point to a specific location in the body of EXITRAP where it will execute when the trap is sprung.

So, when the hooked application subprogram tries to RETURN to its caller, things happen pretty much as they normally do--the parameter list is deleted, the Q and S registers point where they are supposed to, and the status bits and X register are properly restored; the radical difference is that the process is now executing somewhere within the body of the EXITRAP procedure!

What the EXITRAP code does at this point is to immediately call segment-internal procedure EXIT_TRAP, generating a "clean" stack marker which establishes a foundation for working storage and preserves the status bits for the original intended return point. EXIT_TRAP goes out to the exit trap stack, reads the clean-up routine's *plabel* from the current entry, and calls it. Upon return therefrom, EXIT_TRAP retrieves the return vector saved in the current entry and writes it into its above stack marker. After deleting the now obsolete entry from the trap stack by

decrementing the entry count in the header, EXIT_TRAP exits, and everything is back to normal.

Totally confused? OK, here's a simplified overview of the action:

Procedure A calls procedure B. Procedure B calls procedure C. Procedure C calls EXITRAP, passing the *plabel* of procedure D. EXITRAP saves procedure B's return vector and gives it a new one, saves procedure D's *plabel*, and exits; procedure C is now running. Procedure C exits; procedure B is now running. Procedure B exits; it is EXITRAP, not procedure A, that is now running. EXITRAP calls EXIT_TRAP. EXIT_TRAP calls procedure D, pursuant to procedure C's request. Procedure D exits; EXIT_TRAP is now running. EXIT_TRAP replaces its own return vector with procedure B's original return vector and exits; procedure A is now running from the point where procedure B was originally supposed to have returned.

EXITRAP checks for three conditions, assisted by O/P procedures that read from PXFIXED: a) The stack marker to be trapped exists and is not the *initial stack marker*, i.e. its Q > Qi+4; b) the calling process has DS capability if there is no available Green area of sufficient size (this is actually done by BLDUTAB); c) the trap-in-progress flag is FALSE (this is set to TRUE by EXIT_TRAP around the call to the clean-up routine).

What If the Clean-Up Routine Is Program-Resident?

Usually, the best place for the clean-up procedure is in the same program unit as the procedure requesting the hook, or some place where it has convenient access to whatever state information needs to be reset. A program-resident procedure would have the most efficient access to everything between DL and Qi; but then, the LOADPROC intrinsic would be no help as far as obtaining its *plabel* for EXITRAP (or anything else with a *plabel* parameter).

In the venerable FORTRAN language, you have recourse to the EXTERNAL statement, which allows your program to bind to any SL- or program-resident procedure without actually calling it. Wherever a symbol declared EXTERNAL appears as an actual argument passed to a subprogram, the compiler generates an LLBL ("Load Label") instruction which extracts the procedure's *plabel* from the appropriate table. In FORTRAN 66, the *plabel* is passed by value; in FORTRAN 77, it is effectively passed by reference, but may be forced to a by-value construction through an $ALIAS directive.

If, however, your primary source language is COBOL, you have a slight problem, but one possible solution may be found in Section V.

IV. Good Things for FORTRAN (and the People Who Use It)

Here are further applications using stack markers.

Internal Subroutines: Emulating PERFORM

A fairly common situation in FORTRAN program design: The same lengthy sequence of statements must be executed at several different places, but it's rather too cumbersome to break them out into a separate procedure, because they need to share a substantial amount of local (Q-relative) data--perhaps, even a FORMAT statement or two. The conventional solution is to code the statement block once, labeling the first statement and ending with a computed GOTO listing the labels of all possible return points. The variable designated in the computed GOTO must be set to index the correct return label before you branch to the statement block. It works--but one wishes one had a construct as simple as COBOL's PERFORM verb, as compatible with the principles of structured programming, and that doesn't make a nightmare out of fixing or updating the program in the future.

We've devised a PERFORM emulator that takes advantage of FORTRAN's unique Alternate Return feature, whereby a subroutine may, at its option, return control via a labeled statement designated by the caller. Here, in brief, is how Alternate Returns work at object code level:

FORTRAN employs a *return path index* corresponding to a label in the subroutine's calling sequence as written; the actual stacked parameter list does not include or reference any kind of label information. A FORTRAN 66 caller zeroes the X register just prior to the PCAL; the callee thus finds a 0 in the stack marker at Q-3 (the old X register word). When executing a "RETURN *n*" statement, the callee's object code sets Q-3 to *n*, corresponding to the *n*th label listed in the caller's source code; then it's up to the caller's object code to branch to the equivalent PB-relative address. FORTRAN 77, on the other hand, communicates the return path index by the same method as for an INTEGER*2 function value; the callee stores the index at Q minus 4 minus the word length of the formal parameter list. No Alternate Return occurs if the caller finds a 0 in the X register, or at the top-of-stack (S-0), as the case may be.

Our PERFORM emulator scheme consists of three procedures:

LOCTOS(TOS)
: Called at the beginning of a (sub)program to snapshot the normal top-of-stack address after the initialization code has allocated local working storage and any pointers to global data; returned to variable TOS.

ISCAL(*$label*)
: Branches to the statement block or internal subroutine at statement *label*.

ISXIT(\TOS\)
Marks the end of a block and determines from TOS (passed by value) whether to continue on to the next sequential statement or return from the block to the statement following the most recent ISCAL call.

In FORTRAN 66 code, they are implemented like this:

```
      INTEGER TOS
      ...
      CALL LOCTOS(TOS)
      ...
      CALL ISCAL($label)
      ...
      CALL ISCAL($label)
      ...
label CONTINUE
      ...
      <statements constituting the internal subroutine>
      ...
      CALL ISXIT(\TOS\)
```

Internally, LOCTOS sets formal parameter TOS equal to address Q-5. ISCAL and ISXIT communicate through an extra word at the top-of-stack containing the PB-relative address of the instruction following the PCAL to ISCAL, the start of the compiler-generated code that handles the Alternate Return logic; this address is provided automatically in ISCAL's return vector.

When ISCAL is called, the stack looks like this:

```
Q-4     Caller's local data.
      +-----------------Stack Marker-----------------+
Q-3   | Return path index; initialized to 0.         |
      +----------------------------------------------+
Q-2   | Return address.                              |
      +----------------------------------------------+
Q-1   | Old status register.                         |
      +----------------------------------------------+
Q-0   | Delta-Q.                                     |
      +----------------------------------------------+
```

With no stacked parameter list[8] the only way to leave an extra word at the top-of-stack for the caller is to make room for it by moving the stack marker; so, ISCAL copies it into local array NEWSM and increments the delta-Q in NEWSM(3) by the offset of that element relative to Q; this repairs the linkage to the caller's above stack marker. The return address for the procedure will be the return address for the internal subroutine; this is copied into Q-3, which is the target post-EXIT top-of-stack. The X register will be restored from NEWSM(0); storing a 1 thereto sets up the emulated "RETURN 1". To activate the relocated stack marker, ISCAL sets the Q register to point to NEWSM(3) by "pushing" its address and ASSEMBLEing a "SETR Q" instruction. If

the address of this element was Q+*d* before the change-over, then the stack decrement (SDEC) for the EXIT must be *d*-1 to clear away ISCAL's local data and other debris, leaving the internal subroutine's return address. The EXIT instruction may be built at the current top-of-stack and executed by ASSEMBLEing an "XEQ (0)"? Then, the 1 "popped" into the X register actuates the Alternate Return to the first statement of the internal subroutine.

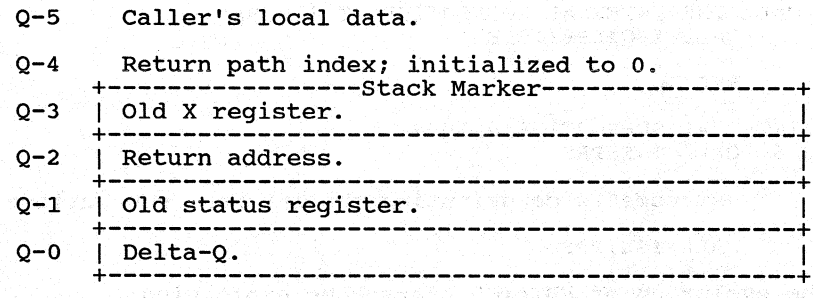
At the other end, when ISXIT is called, the stack looks like this:

```
Q-6     Caller's local data.

Q-5     Caller's local data OR return address for the
        internal subroutine, depending upon--

Q-4     Pointer to normal top-of-stack (TOS by value).
      +-----------------Stack Marker-----------------+
Q-3   | Old X register.                              |
      +----------------------------------------------+
Q-2   | Return address for the procedure.            |
      +----------------------------------------------+
Q-1   | Old status register.                         |
      +----------------------------------------------+
Q-0   | Delta-Q.                                     |
      +----------------------------------------------+
```

ISXIT first compares address Q-5 with the TOS value presumably returned previously by LOCTOS. If Q-5 does not exceed TOS, ISXIT immediately returns without doing anything; this arrangement allows you to execute the statement block just like a COBOL paragraph, which you can GOTO, drop into or "PERFORM".

If Q-5 > TOS, ISXIT copies the internal subroutine's return address from Q-5 to Q-2, making it <u>its</u> <u>own</u> return address. Loading a 0 into Q-3 sets up an emulated normal RETURN as if from ISCAL. ASSEMBLEing an "EXIT 2" deletes the extra word generated by ISCAL, restoring the stack to its former condition, and brings us back precisely to where ISCAL returned previously; this time, the 0 in the X register directs the flow to the next sequential statement.

A fourth procedure, ISCAL_77, provides for FORTRAN 77's use of the top-of-stack instead of the X register to hold the return path index. When the statement "CALL ISCAL_77(**label*)" is executed, the stack looks like this:



ISCAL_77 does the same thing as ISCAL, except that it copies the return address into Q-4 with bit 0 set on, and stores the "RETURN 1" trigger in Q-3; after the exit, the return path index is popped off the stack by the Alternate Return code, exposing the return address for the internal subroutine.

As you may have guessed, bit 0 is used to indicate to ISXIT which Alternate Return M.O. prevails. If it is set, ISXIT copies bits (1:15) of Q-5 into Q-2, loads a 0 into Q-5 and does an "EXIT 1", leaving the extra word to be deleted by the Alternate Return code.

When Internal Subs Aren't the Only Game in Town

Using the top-of-stack to carry the return point address allows internal subroutines to be nested. For this very reason, however, ISCAL_77 must be used with considerable caution, since FORTRAN 77 also uses the top-of-stack to manage DO loops and IF-THEN-ELSE constructs; in a recent MANMAN modification, I found this to be extremely limiting.

The following bit of FORTRAN 77 code illustrates the solution I've worked out:

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT
      ...
      INTEGER GETXREGF,ISPBA,CCX,I
      ...
C FIND CODE ADDRESS OF INTERNAL SUBROUTINE AT
C LABEL 1000.
      CALL FMTATOP(ISPBA)
      CALL SAVECCODE
      CCX=GETXREGF()
C IF CCG FROM FMTATOP, GO EXECUTE HEADER ROUTINE.
      IF(CCX.GT.0)GOTO 1000
C WHEN WE'RE HERE, CODE ADDRESS IS IN ISPBA.
      ...
C CALL THE INTERNAL SUBROUTINE AT 'ISPBA'.
      CALL ISCALPB(ISPBA)
      ...
      DO I=1,10
```

```
      ...
C CALL THE INTERNAL SUBROUTINE AT 'ISPBA'.
      CALL ISCALPB(ISPBA)
      ...
      END DO
      ...
C INTERNAL SUBROUTINE HEADER.
1000  CALL ISENTRY
      ...
      <statements constituting the internal subroutine>
      ...
      CALL ISXITPB
```

The evolution of FMTATOP bears some explaining:

For the INTEREX '91 Conference, I wrote a piece called "MPE V/E FORTRAN: The Internals of Alternate Return Paths" (Paper #3212), wherein, as a sidelight, I described how I managed to write--in FORTRAN 77--a general interface to the FORTRAN 66 Formatter. FMTATOP and its original dance partner FMTABOT were my solution to the problem of obtaining the PB-relative address required as a parameter by the FMTINIT' procedure[10]--a lead-pipe cinch to do in SPL, but highly problematical elsewhere! Since this earlier paper is of interest only to hard-core FORTRAN *afficionados* (of whom I seemed to be the only one in San Diego that week), I would just as soon you did not have to dig it out to learn how the FMTATOP/FMTABOT *pas de deux* works.

FMTATOP loads its return address into variable ISPBA with bit 0 set on, sets up CCG and does an "EXIT 0"; on finding CCG, the caller would be expected to branch directly to the call to FMTABOT, strategically inserted between the call to TFORM' and its own Condition Code test. FMTABOT, being itself parameterless, would find the address of ISPBA at Q-4. On finding ISPBA(0:1)=1, it would swap the contents of ISPBA and Q-2 with bit 0 cleared, set up CCE and "EXIT 0"; the caller would then be executing from FMTATOP's Condition Code test, with ISPBA pointing to TFORM''s Condition Code test and ready to pass by value to FMTINIT'. The ISPBA address would be left at the top-of-stack, so that when the call to FMTABOT was dropped into upon return from TFORM', at which point the Formatter's save area would have been cleared away, ISPBA(0:1)=0 would signal an immediate vanilla exit, allowing TFORM''s Condition Code to pass through "like green corn through the new maid."[11] To avoid any unpleasant surprises at run time, a separate procedure would have to be called at an appropriate juncture to delete the ISPBA address from the stack.

ISENTRY is an entry point in FMTABOT which serves the same purpose, except that it returns (with CCE to FMTATOP's Condition Code test) via "EXIT 1", deleting ISPBA's stack address, with the understanding that, like FMTATOP, it is called in the course of housekeeping <u>once</u> <u>only</u> per internal subroutine; label 1000 must be reachable <u>only</u> on CCG from FMTATOP, and from nowhere else in the code. The code segment address saved in ISPBA points direct-

ly to the next statement after the ISENTRY call, the start of the main body of the internal subroutine.

The SAVECCODE and GETXREGF procedures were developed to work around the "IF(CCODE)..." internals which are not compatible with any procedure not declared a SYSTEM INTRINSIC. SAVECCODE determines the numeric equivalent (0, +1 or -1) and hands it off to GETXREGF through the X register.

ISCALPB and ISXITPB almost exactly emulate the SCAL and SXIT instructions and don't care where the normal top-of-stack is supposed to be. The advantage they have over ISCAL_77/ISXIT in a DO loop or IF-THEN-ELSE situation is that the compiler is not aware of your intention to branch outside the range of the block and back in at the next statement, so there is no conflict with block management stack operations.

Functions Within Subroutines

Both FORTRAN compilers are in full agreement on this point: If a subprogram has multiple entry points, then these entry points must be either all subroutines or all typed functions. Now, supposing you have a subroutine that holds state information initialized in DATA statements, and the need arises for a function for convenient use in expressions that returns the current value of one of these state variables. Under the conventional rules, your function would have to be in a separate program unit, which means that the state variable in question would have to be moved to a COMMON block, which would in turn have to be initialized in yet a third unit called a "BLOCK DATA" subprogram. Wouldn't it be much simpler just to have a secondary entry point that could be called as a function? Not only can this be done unbeknownst to the compiler, but the function return operation itself can be encapsulated in a general procedure!

For entry points serving non-character functions, we have

```
                1
FUNCRTN[{2}](ITEM,SIZE)
                3
```

where ITEM is the value to be returned, and SIZE indicates the size of the datum in words as a power of 2. Exit from a function embedded in a SUBROUTINE unit is through a call to FUNCRTN instead of the RETURN statement. FUNCRTN has three additional ports, numbered 1-3, so that calls involving ITEMs of different type may coexist in the same unit. After copying ITEM into the function return area allocated by the caller of the function entry point, FUNCRTN sets the Q register to Q'=Q-'Q-0' and EXITs with the appropriate SDEC.

Now, how on earth does FUNCRTN, being a general procedure, know where to set the function value and how many parameter list words to delete on the way out? Well, since secondary entry points are

permitted to have parameter lists different from the primary's and from each other's, and since every RETURN statement serves the entire subprogram unit, then for each point of entry, initialization code stores the current parameter list size at Q+1. The object code emitted for each RETURN statement reads the SDEC from Q+1 and dynamically exits via the top-of-stack, in the manner described above.

So, FUNCRTN reads the SDEC from Q'+1 and stores the function value at Q'-3-SDEC-2**SIZE(14:2).

Type CHARACTER functions require a somewhat different strategy. In FORTRAN 66 mode, the caller of a character function allocates a return area according to the declared size of the string to be returned. It then pushes a byte pointer to be used by the callee to locate the return area. It is the responsibility of the callee to know the length of the return string and blank-fill as needed. The callee is also expected to leave the return area pointer at the top-of-stack.

In FORTRAN 77 mode, basically the same events occur, except that: 1) The return area is pointed to by a 2-word *descriptor*--byte pointer plus positive byte count; 2) the callee is expected to delete the return area descriptor along with its parameter list.

In both cases, we cover the return area vector in the ENTRY statement by inserting a suitably-sized type CHARACTER variable at the beginning of the formal argument list; this variable may be referenced just like any garden-variety argument. For FORTRAN 66 mode, we exit by calling CHAR66FUNCRTN, which sets the SDEC to one less than the parameter list length, so that the byte pointer is left at the top-of-stack as required. But for FORTRAN 77 mode, no special procedure call is needed--a vanilla RETURN statement does the job quite nicely!

Needless to say, when writing any subroutine-type subprogram with an entry point to be referenced as a typed function, the formal checking level must be 0.

V. "..., But We Only Have COBOL!" -- Finding the *Plabel* of a Non-SL Subprogram

Have you ever tried in a COBOL program to use one of the error-trapping intrinsics (XARITRAP, XLIBTRAP, etc.) or the IPC software interrupt feature, or arm the CONTROL-Y break, when your intended trap-handling subprogram does not reside in an SL?

Why would you want it to reside in the program? Let's assume that you're mainly interested in CONTROL-Y. Like the BREAK key, CONTROL-Y can interrupt your program at any point--in the middle of a paragraph, in the middle of a sentence, anywhere. If you want more control over where or when the processing occasioned by CONTROL-Y can take place, your trap handler must do nothing more than set a flag that your main program can test periodically to

determine if CONTROL-Y has been sensed. By far the best place for such a flag is in the global stack area. COBOL'85, whose unique features are available through the COBOL II compiler's ANSI85 entry point, has an EXTERNAL clause which, when applied to a level 01 data item, makes it equivalent to a labeled COMMON block in FORTRAN, thus enabling two or more subprograms to share statically-mapped data items without relying on CALL-USING and a LINKAGE SECTION.

Now, here's the dilemma: A COBOL II subprogram using the EXTERNAL keyword, regardless of whether it's dynamic, cannot reside in an SL for the same reason that a FORTRAN subprogram using COMMON can't. This precludes the possibility of it being LOADPROCed; but even COBOL'85 provides no other way to come up with that one object without which you cannot do business with any trap-arming intrinsic--namely, the trap handler's *plabel*, as distinct from its name in 15 bytes or less.

If you had a FORTRAN or SPL compiler on your system, you'd be able to get out of this bind fairly easily by writing a highly-efficient support procedure--you could even write a "BLOCK DATA" subprogram to pre-initialize the EXTERNAL area!--but this whole section is based on the hypothesis that COBOL II is all you've got. I concede that even under this constraint it is theoretically possible for an SL-resident trap handler to communicate through a heap location normally used by a different 3GL, but such encroachment upon another subsystem's space, even if it isn't around to enjoy it, should be avoided on principle.

For a program-resident trap handler, there's a solution more respectful of subsystem boundaries--consider the following:

* The *plabels* of program segment entry points are the same for every process sharing the program, because of the way that segment numbers are assigned at process-creation time. Their values are thus effectively fixed at preparation time and can be predicted from examination of the PMAP.

* When a program is PREPped, every structurally sound segment in the USL, regardless of whether any entry point therein satisfies an external reference, becomes part of the program, and is loaded whenever the program is loaded.

A *plabel* is not a hash code derived from an entry name; its 16 bits are neatly divided into the following fields:

(0:1) Mapping flag (MF); qualifies the segment number in the lower 8 bits. Set to 1 for an MPE segment in the system library; set to 0 for all other code segments.

(1:7) STT (Segment Transfer Table) index; points to the entry point address in the STT of the target code segment.

(8:8) Segment number. If MF=1, this is the actual CST index of an MPE segment; otherwise, it is the "local" num-

ber[12] (LCL#) whereby the target segment is referenced by the current process.

As I stated in *S-II*, when a program is prepared, "logical" segment numbers (LOG#s) are assigned consecutively from 0 with no gaps, and when a process sharing the program is created, LCL#s are assigned first to the program segments beginning with 1 for LOG# 0, so that for every program segment in every sharing process, LCL# = LOG# + 1; therefore, for each entry point in a program segment, the segment number field in its *plabel* is constant for all sharers.

The STT is a physical component of the prepared code segment; so, the STT index field must also be constant for each program-resident entry point in all sharing processes.

The MF field, of course, will always be 0, since it can only be set to 1 for a system library segment.

Now, how does all this help you to reconstruct a *plabel* in a COBOL-only shop?

The good news is, you don't need any special capabilities; in fact, you'd be doing essentially what COBOL'85 object code does in the case of a CALL statement with an EXCEPTION clause, to determine if the called subprogram resides in the program file. The bad news is, it's a non-trivial task, and in terms of processing time, also fairly expensive; it should be done once only for each required *plabel*, as part of housekeeping.

Success depends upon your program being PREPped with the ;FPMAP option. This embeds in your program file a "machine-readable" PMAP which can be accessed through documented system intrinsics; these, and the FPMAP record formats, are clearly described in the Segmenter manual. A second requirement is that all personnel who need to run your program have READ and LOCK access to the program file from their logons.

The FPMAP record for your trap handler's entry point is obtained by calling FINDPMAPNAME, passing the name of the target entry point as an ASCII string. This record tells you half of what you need to know--the LOG#, which effectively gives you the LCL# for the *plabel*; the half it doesn't tell you is the STT index. What you have to do now is to open a scratch file, call DUMPPMAP passing the LOG# obtained from FINDPMAPNAME, and initialize a counter on your stack to 0.

Entry point records come out of DUMPPMAP sorted by LOG# by STT index, and cover both revealed and hidden entry points[13] A code segment can have a maximum of 127 entry points total, all with unique names, so your dump file will contain at least 2, but not more than 128, records, the first of which is the only Type 0 (segment header) in the dump.

Sequentially read the dump file beginning with record 1,

incrementing the counter by 1 for each read. When your target entry name matches either the "Procedure Name" in a Type 1 record, or the "Secondary Entry Point Name" in a Type 2 record, the STT index is simply the current value of the counter.

Your trap handler's *plabel* is now fully constructed and may be either used immediately or saved for later.

The best way to write and arm a CONTROL-Y routine is explained in Section VII, but don't read it just yet--there are some concepts that you need to understand first, and these are introduced in the next section.

VI. "..., But We Don't Have SPL!" -- Accessing the "Heap"

The stack region between DL and DB--also known as the "heap", or "user-managed" global area--is used by a number of subsystems to hold information to be shared among several SL procedures and retained between calls. It may be allocated either through the :PREP or :RUN command, or programmatically by calling the DLSIZE intrinsic. No special capabilities are needed to allocate and access this area, but since no program variables can be mapped there at compilation time, it can only be addressed dynamically in terms of negative integers, or arithmetic expressions with a negative integer result, all addresses being relative to DB.

The first 12 words below DB are reserved for supported subsystem use. FORTRAN 77 uses DB-12 to point to its library globals, which are stored entirely in the heap.

On MPE V systems, only SPL programs can manage the heap self-sufficiently, but the SPL compiler is not "standard equipment". For access from other 3GLs, you must build three basic support procedures--one of which, in the absence of SPL, requires patching of its Relocatable Binary Module (RBM) before it can be installed.

Procedure GETDLREG

It is not easy to access the heap on Classic systems without budgeting extra money for an SPL compiler--not even a documented intrinsic for checking the current heap size, i.e. the current setting of the DL register, without changing it. For instance, you cannot check for the existence of a FLUT[14] without either possibly allocating heap unnecessarily or risking a "BOUNDS VIOLATION" trap. When no heap is allocated, DL is coincident with DB, so any attempt to address DB-1 would actually be addressing DL-1--in effect, pointing into the PCBX which is strictly off-limits in User Mode.

Here's how we've solved the problem in our non-SPL shop:

```
:FTN DLREGS,DLREGU;INFO="USLINIT"

PAGE    1   FORTRAN 77/V    HP32116A.01.00  COPYRIGHT HEWLETT-PACKARD CO. 1983
  TUE, NOV  3, 1992,  4:47 PM


   0     0         $USLINIT

   0     1.000     $CONTROL STANDARD_LEVEL SYSTEM,SHORT,SEGMENT "DL'NEG"
   0     1.100     $TABLES,CODE_OFFSETS,LIST_CODE ON
   0     2.000     C
   0     3.000     C GET DL REGISTER
   0     4.000     C
   1     5.000           SUBROUTINE GETDLREG(K)
   1     6.000     C
   1     7.000     C DUMMY STATEMENT - ALLOCATE 2 WORDS
   2     8.000           IX=IX <--- statement 2
   3     9.000           RETURN
   4    10.000           END

Name            Class           Type          Offset        Location
----            -----           ----          ------        --------

GETDLREG        Subroutine
IX              Variable        Integer*2     Q+2           Local
K               Variable        Integer*2     Q-10,I        Argument


     C O D E   L I S T

    0 000000 .. NOP, NOP
    1 000000 .. NOP, NOP
    2 035001 :. ADDS 1
    3 041401 C. LOAD Q+1 \ compiled from statement 2;
    4 051401 S. STOR Q+1 / see "CODE OFFSETS".
    5 140001 .. BR   P+1
    6 031401 3. EXIT 1


                  C O D E   O F F S E T S

STMT  P-LOC    STMT  P-LOC    STMT  P-LOC    STMT  P-LOC    STMT  P-LOC
   2 000003       3 000005       4 000006


   NUMBER OF ERRORS =     0   NUMBER OF WARNINGS =     0
   PROCESSOR TIME   0: 0: 0   ELAPSED TIME      0: 0: 9
   NUMBER OF LINES =     11
```

```
END OF COMPILE
:
```

As you can see, this little program does not work as written; the purpose of the meaningless statement "IX=IX" is to allocate 2 words in the RBM where you may surgically insert the machine instructions needed to return the current DB-relative address of DL through formal parameter K. To make it relatively easy to locate procedure-relative locations 3 and 4, you should compile it into a separate USL, from which you may copy the patched RBM into the USL of your choice.

The required instructions are:

PSHR DL	(%024440)	Push the DL register value to the top-of-stack.
STOR Q-4,I	(%053604)	Pop the value at the top-of-stack into the address given in Q-4, i.e. the address of formal parameter K.

(Don't let the mapping in the "TABLES" section confuse you; these offsets are expressed in terms of bytes.)

To do the patch, you must run DISKED5 and use the >FILE command. All USL file sectors are gathered for you and assigned logical sector numbers relative to the file label. A USL is so constructed that each record is equivalent to a sector, so absent any user labels, record *N* will be in logical sector *N*+1.

First, >DUMP record 0 to get the SAI vector to the INFO block containing your isolated target RBM; this is in 2-word format in words %14 and %15, with the record number in the upper 25 bits, and the word offset in the lower 7. Add 1 to the record number to obtain the logical sector to >DUMP next; therein you will easily recognize the binary code displayed by the $LIST_CODE directive, and where you must make the change.

Integer Procedure PEEK

This FORTRAN 77 function reads one word of data from any calculated stack address (subject to bounds-checking):

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT
$CHECK_FORMAL_PARM 2
C
C RETURN THE CONTENTS OF A DB-RELATIVE ADDRESS.
C
      INTEGER FUNCTION PEEK(ADDR)
      INTEGER ADDR
C
      PEEK=ADDR
      RETURN
```

```
      END
```

I'm not kidding! This works precisely as written--provided that you call it this way:

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT
$ALIAS PEEK (%VAL)
      ...
      INTEGER PEEK
      ...
C LOCATE I/O GLOBALS.
      IOB=PEEK(-12)
```

Thus, any integer passed by value is formally interpreted as a data reference!

Procedure POKE

This FORTRAN 77 subroutine stores one word of data to any calculated stack address:

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT
$CHECK_FORMAL_PARM 2
C
C SET THE CONTENTS OF A DB-RELATIVE ADDRESS.
C
      SUBROUTINE POKE(DATUM,ADDR)
      INTEGER DATUM,ADDR
C
      ADDR=DATUM
      RETURN
      END
```

Usage:

```
$CONTROL STANDARD_LEVEL SYSTEM,SHORT
$ALIAS POKE (%REF,%VAL)
      ...
      INTEGER DATUM,ADDR
      ...
C SET VALUE.
      CALL POKE(DATUM,ADDR)
```

The usefulness of PEEK and POKE is entirely in the latitude the compiler allows you in constructing the actual parameters. They work because compilers don't care what your game is, as long as you play by their rules.

Different 3GLs offer different syntax in support of value parameters. FORTRAN 66 and COBOL II use back-slashes (\) to bracket them in procedure calls; FORTRAN 77, on the other hand, relies on $ALIAS directives which are a real pain and all too easy to forget to remember not to forget.

The best way to ensure that PEEK and POKE always get the caller coöperation they need is to make them declarable as *intrinsics*. This is indeed possible without the SPL compiler, for standard equipment on Classic systems is the compiler for SPL intrinsic headers, BUILDINT.PUB.SYS, which writes a compiler-readable procedure definition into a SPLINTR (SPL INTRinsic) file.

An intrinsic does not itself have to be written in SPL, nor does it have to reside in the system library; other than to write, compile and install the procedure code as you normally would, all that's needed is to write enough SPL code for the intrinsic header and run BUILDINT.

The respective headers for PEEK and POKE would look like this:

```
INTEGER PROCEDURE PEEK (ADDR);
VALUE ADDR;
INTEGER ADDR;
OPTION EXTERNAL, CHECK 2;

PROCEDURE POKE (DATUM, ADDR);
VALUE ADDR;
INTEGER DATUM, ADDR;
OPTION EXTERNAL, CHECK 2;
```

For full details on writing and implementing non-SPL intrinsics, see my paper "Designing an INTRINSIC Procedure in Your Favorite Language"[15]

How 'Bout Some Icing on the Cake?

The principles involved in producing PEEK and POKE may be easily applied to writing procedures for copying arrays of data from one stack area to another. This may be done by writing a single program unit with four entry points, or "ports", with identical formal parameters, covering respectively: reference-to-reference; reference-to-pointer; pointer-to-reference; pointer-to-pointer. Appropriate SPLINTR definitions are all that would be needed to functionally differentiate the ports.

I will now show you one more procedure that will give you more complete access to your stack.

Integer Procedure QREG

This one gives you the location of your above stack marker; it can be written entirely in FORTRAN 77 without patching the RBM, but must be compiled twice.

```
:FTN DEMO01S,$NEWPASS;INFO="CODE OFFSETS,LIST CODE"

PAGE    1   FORTRAN 77/V    HP32116A.01.00  COPYRIGHT HEWLETT-PACKARD CO. 1983
  WED, DEC  9, 1992, 12:37 PM


   0     0         $CODE_OFFSETS,LIST_CODE

   0     1.000     $CONTROL STANDARD_LEVEL SYSTEM,SHORT,FTN3000_66 CHARS
   0     2.000     C
   1     3.000           INTEGER FUNCTION QREG()
   1     4.000     C
   1     5.000     C RETURNS CURRENT Q REGISTER VALUE.  "LOOK, MA--NO SPL!"
   1     6.000     C
   2     7.000           INTEGER REF
   3     8.000           SYSTEM INTRINSIC PEEK
   3     9.000     C
   3    10.000     C Q-RELATIVE MAPPING OF REF (FROM 1ST PASS COMPILE).
   4    11.000           REF=0  <--- statement 4
   4    12.000     C LOCATE REF.
   5    13.000           LOC=BADDRESS(REF)
   5    14.000     C CALCULATE Q FROM REF LOCATION.
   6    15.000           IQ=ISHFT(LOC,-1)-REF
   7    16.000           QREG=IQ-PEEK(IQ)
   8    17.000           RETURN
   9    18.000           END

     C O D E   L I S T

    0 000000 .. NOP, NOP
    1 000000 .. NOP, NOP
    2 035006 :. ADDS 6
    3 000600 .. ZERO,NOP \ compiled from statement 4;
    4 051404 S. STOR Q+4 / see "CODE OFFSETS".
    5 171404 .. LRA  Q+4
    6 010201 .. LSL  1
    7 051406 S. STOR Q+6
   10 041406 C. LOAD Q+6
   11 004500 .@ DUP, NOP
   12 051403 S. STOR Q+3
   13 010301 .. LSR  1
   14 101404 .. SUBM Q+4
   15 051402 S. STOR Q+2
   16 000600 .. ZERO,NOP
   17 041402 C. LOAD Q+2
   20 000000 .. NOP, NOP
   21 024410 ). PSHR ST,
   22 026542 -b EXF  6:2
   23 051405 S. STOR Q+5
   24 101402 .. SUBM Q+2
   25 002400 .. NEG, NOP
   26 051401 S. STOR Q+1
   27 140001 .. BR   P+1
   30 041401 C. LOAD Q+1
   31 051604 S. STOR Q-4
```

```
   32 031400 3. EXIT 0


 PAGE    2


                        C O D E   O F F S E T S

STMT  P-LOC     STMT  P-LOC     STMT  P-LOC     STMT  P-LOC     STMT  P-LOC
   4 000003        5 000005        6 000013        7 000016        8 000027
   9 000030


     NUMBER OF ERRORS =      0   NUMBER OF WARNINGS =     0
     PROCESSOR TIME   0: 0: 1   ELAPSED TIME       0: 0:20
     NUMBER OF LINES =      18


END OF COMPILE
:
```

Local variable REF is selected as a reference point. In this initial pass, it is arbitrarily set to 0, but for the second pass, the source code must be edited to set REF to its own offset relative to Q, once that is determined from the first-pass printout. As you can see from the instructions generated from statement 4, REF is mapped to Q+4.

(Why not use the more readable $TABLES directive to locate REF? Well, this may shock you, but if you're using compiler version A.01.00, you literally cannot believe everything you read in the "TABLES" listing! But you can always believe the object code.)

Since BADDRESS always returns a byte pointer, the result must be right-shifted one bit before subtracting the REF offset to obtain the local Q; from which you then subtract the delta-Q saved in Q-0 to obtain the caller's Q, using your home-grown PEEK intrinsic.

QREG is obviously quite useful if you're interested in tracing back through your stack markers. For that purpose, you would need one more procedure for pin-pointing the location of the initial stack marker, i.e. the above stack marker for the Outer Block, from any hierarchical level in your program. Here is yet another case where the most reliable and efficient method requires privilege. Tables Chapter 7 and the procedures I've described in this section provide you with all the tools you need to read Qi from PXFIXED, but since you have no recourse to SPL, emulating "OPTION PRIVILEGED" requires some patching. The USL file structure is clearly enough described in Tables Chapter 9 that you can, without too much difficulty, locate your procedure's Type 4 entry in the Directory and set its "I" (Privileged) bit with DISKED5 (just make sure the "A" bit in the entry you've found is off if you've compiled into this USL more than once).

QREG also paves the way for an unusual technique that I have used successfully and that greatly facilitates accessing arrays in the heap. This is the technique of dynamically remapping local arrays, and I will use FORTRAN 66 as my vehicle for explaining it.

A local array is always mapped by the compiler indirectly to a static location relative to Q, so if you have in your "data division" (so to speak) the type declaration "INTEGER DBUF(*dim*)", and you compile with the MAP option, you will find in the Symbol Map this line:

```
DBUF                INTEGER     ARRAY        Q+%xx,I
```

[*xx* => some octal number representing words.]

This means that prior to the first executable statement in the program, the initialization code stores the base address of array DBUF at location Q+%*xx*. In FORTRAN 66, this is actually the address of what would be the 0th element, to which a subscript multiplied by the element size is added to obtain the correct element address. To use DBUF to symbolically reference an integer array of any length in the heap, you must first call QREG to find your local Q, then subtract 1 from the base address of the heap array and POKE the result to Q+%*xx*. Thereafter, whenever you reference an element of DBUF in an expression, or assign a value to an element, you will be reading from, or writing to, the array in the heap instead of the one that was allocated locally, and with the same degree of efficiency.

As with QREG, implementing this technique is a two-pass compilation process; the first pass to obtain the correct pointer address from the Symbol Map, the second with the correct Q-relative offset appropriately placed in your code.

VII. Arming CONTROL-Y from COBOL II

If you've read the Intrinsics manual on how to use XCONTRAP, you'd understand the peculiar difficulties of writing a CONTROL-Y trap procedure in any language but SPL. But with the previously-described heap access procedures in hand, it is possible to emulate FORTRAN's "ON CONTROLY CALL" statement in COBOL II. Here's how:

First, make sure that Compiler Library procedures F'CONTRAP and F'CONTRAPPROC are installed in your system library. Even if COBOL II is your only available 3GL, these procedures should be present since they are bundled in the same system code segment (CLIB'01) with procedures required by COBOL II and RPG. Also, the PEEK and POKE procedures can be written in COBOL II and called "at crossed purposes" in the same manner; indeed, these are the only two heap procedures you will need, since the object code should have already allocated at least the first 5 words

below DB (you can always call DLSIZE if you're not sure).

The Tables manual to the contrary notwithstanding, DB-3 is reserved for FORTRAN, not COBOL. For a FORTRAN program containing an ON statement, it points to a 17-word TRAPCOM' block in the global stack area used to save trap handler *plabels* according to error type. To share or dynamically allocate this resource, either your main program or a non-dynamic subprogram must first define in WORKING-STORAGE a 17-word array initialized to all zeroes. Then, if a PEEK of DB-3 returns a -1 (no TRAPCOM' allocated), you must obtain the array address with .LOC. and POKE it to DB-3.

Your CONTROL-Y trap handler may be written as a vanilla COBOL II subprogram with no LINKAGE SECTION. To arm it, obtain its *plabel* and pass it by value to F'CONTRAP:

```
77  TRAP-PLABEL      PIC S9999 USAGE COMP.
...
    CALL "F'CONTRAP" USING \TRAP-PLABEL\.
```

The actual trap procedure that is armed is F'CONTRAPPROC, which calls your trap handler and calls RESETCONTROL after your handler GOBACKs. It may be disarmed by calling XCONTRAP, but can only be re-armed by calling F'CONTRAP (not strictly true, but true for most practical purposes).

F'CONTRAP saves your trap handler's *plabel* in TRAPCOM' array element 11 (address DB+'DB-3'+10); you can switch to a different trap handler at any time from any subprogram simply by changing the *plabel* stored at that location.

Oh yes, don't forget to open a file on $STDIN or $STDLIST and call FCONTROL 17 to enable "subsystem BREAK".

Do I really have to call FOPEN just for that?

Well now, come to think of it, you don't, actually--but FCONTROL requires an MPE file number, and you may not have a suitable one at the ready. Here's another good use to which an O/P procedure can be put.

Have you ever wondered why it is that no matter when you FOPEN a file, you never get a file number less than 3? It is because numbers 1 and 2 are already in use! Even before your program reaches the first executable statement, MPE has already opened your $STDIN and $STDLIST--on channels 1 and 2, respectively; indeed, these are the file numbers used by the READ(X) and PRINT intrinsics. The problem is, you are not allowed to use these numbers yourself in User Mode--probably to help prevent you from accidentally FCLOSEing them.

You can easily create an O/P procedure to enable CONTROL-Y by calling FCONTROL 17 directly on MPE file number 1, thus saving valuable time and resources.

VIII. FORTRAN Logical Units: Resolving a Compatibility Issue

If you are an ASK MANMAN shop that has upgraded from Version 5.1 or earlier to Version 6.0 or later, then you may be an odd-ball shop like ours that has, and uses, both FORTRAN 66 and FORTRAN 77. In my view, they are different 3GLs, each with its own advantages, and as distinct from each other as they are from COBOL; yet, they are sufficiently similar to greatly facilitate conversion of source code from one to the other.

Certain language constructs shared in common by the two FORTRANs are mutually incompatible because of different internals. One major area of incompatibility concerns the use of Logical Unit (LU) numbers in file operations. For example, LU 1 in a FORTRAN 77 Outer Block does not necessarily refer to the same MPE file number, or even the same file, as LU 1 in a FORTRAN 66 subroutine in the same program. The reason is that each FORTRAN maintains on your stack its own separate FLUT through its own separate set of library procedures. Let's take a look at each one--

The 66 FLUT

The Compiler Library manual describes a somewhat primitive structure. Each FLUT entry consists of a single word containing a unique LU number in the upper byte, and either 0 or an MPE file number in the lower. The physical end of the table is delimited by a word with the number 255 in the upper byte. Since valid LU numbers are in the range 1-99, the maximum size of the FLUT is 100 words.

The FORTRAN 66 FLUT is supported by the code management system. When a program is prepared, the Segmenter builds in the PROG file an initialized image of the DB-Qi stack region, or "system-managed" global area; this includes a FLUT structure for all LU numbers declared in the program. The stack address of the FLUT is set in record 0 so that at process-creation time, the Loader allocates enough heap space to copy the FLUT address to DB-1, whose default setting is -1. It is plain to see why expression of LU numbers as constants is disallowed by the Segmenter in SL procedures, since they are sharable by non-FORTRAN programs.

The FLUT, however, does not have to be engraved in stone at compile time. Using the heap access procedures described in Section VI, you can, from any 3GL, dynamically transfer an existing FLUT into, or build one from scratch in, an array in either global region, or even in the Outer Block's local stack; this array can be pre-extended to the full 100 words to allow for dynamic addition of LU numbers as needed. The Formatter doesn't care where the FLUT is, as long as it is properly pointed to at DB-1; the only question is, how to get the new FLUT address to POKE in there--in FORTRAN 77, you can use BADDRESS in conjunction

with ISHFT; in FORTRAN 66, you'd have to use something like the GETVAL support procedure described in "Designing an INTRINSIC etc.".

The 77 FLUT

All of the globals shared by FORTRAN 77's I/O procedures are managed in the heap, and are allocated as and when needed; the main pointer is set in DB-12, whose default setting is 0. This makes the FLUT inherently dynamic and expandable, so that LU numbers and all I/O statements and generic procedures may be used freely in an SL procedure; they are not recognized as such by the code management system.

Unlike the other FLUT, this one provides for LU 0, and LU numbers greater than 99.

Starting at word 8 of the globals area (address DB+'DB-12'+8) is a block of 16 chain heads. Each one consists of either a link (DB-relative address) to a 54-word FLUT entry, or %77777 indicating an empty chain. Empty and non-empty chain heads are interspersed in the block in no particular order. Word 7 of the globals area points directly to the most recently allocated FLUT entry.

From what I've made out so far, here is a partial list of the FLUT entry contents:

0	MPE file number. If this LU has been closed through a CLOSE statement or UNITCONTROL call, this field contains %77777; the entry may also be removed from the chain.
2	LU number.
8	Link to the next entry in the current chain; %77777 = end of chain.
9½ - 52	File name, left-justified with blank padding.

(Incidentally, this is similar to the way the accessor table is organized in a TurboIMAGE DBG control block.)

FLUT Management Procedures

Understanding of the two FLUT structures should help you to design procedures to synchronize LU assignments in mixed-FORTRAN programs--and you may very well end up with one if you are modifying an extremely large FORTRAN 66 program, and the compiler hands you the software engineer's worst nightmare, the "SYMBOL TABLE OVERFLOW" error.

Armed with heap access procedures, FORTRAN 77 provides the most

convenient access to both FLUTs. Here are some things you should know about the generic FLUT management procedures common to both FORTRANs:

Generic Name	Actual 66 Name	Actual 77 Name ($SHORT)	Actual 77 Name ($LONG)
FNUM	FNUM	F77_FNUM	FTN_FNUM
FSET	FSET	F77_FSET	FTN_FSET
UNITCONTROL	UNITCONTROL	F77_UNITCONTROL	FTN_UNITCONTROL

In FORTRAN 66, the FLUT is assumed to be pre-existent and non-dynamic; therefore, any passed LU number must be already defined on penalty of a library error trap. In FORTRAN 77, an LU number is dynamically allocated if not currently existent.

In FORTRAN 77 code, it is best to use the $SHORT directive; the calling sequences and parameter structures would thus be the same for both sets. The FORTRAN 66 procedures may be referenced as declared SYSTEM INTRINSICs while referencing the native procedures by their actual names. Alternatively, you may modify the FORTRAN 66 procedure references with $ALIAS directives and use the generic names for the native procedures.

Hooking FORTRAN 66 I/O Statements

Another way to synchronize LU assignments in a mixed-FORTRAN program is to hook all FORTRAN 66 I/O statements and externals so that only the FORTRAN 77 FLUT is used. This is made possible by the fact that the FORTRAN 66 Formatter accepts MPE file numbers, in negated form to distinguish them from LU numbers.

In his INTEREX '90 paper "Making Other People's Programs Do What They Were Never Intended to Do--The 'Hook Under the Hood'"[16], Eugene Volokh showed us how to hook calls to system intrinsics by writing intermediary procedures with identical names and calling sequences, from which the originals are called by LOADPROC and *plabel*, and planting them at higher binding levels, usually in account or group SLs as required. One procedure that must be hooked is FMTINIT', which is called transparently by the object code generated by READ, WRITE, ACCEPT and DISPLAY statements. Here, alas, is where the intermediation method of intrinsic-hooking breaks down; if you tried hooking FMTINIT' the same way that you would hook FREAD, you would be likely to get some weird results--for two reasons. One is that FMTINIT' generates temporary global storage at the top-of-stack for shared use by the several compiler-generated procedure calls--IIO', SIO', etc.--that transfer list elements, ending with a call to TFORM' which completes the formatting operation and tries to restore the stack to its original condition. As soon as your FMTINIT' hook returned to the caller, these temporary globals would be deleted from the stack, but still be pointed to at DB-2; the other Formatter procedures could end up clobbering the globals with their own

local stuff. Even if your process made it as far as calling TFORM', who knows what might then happen to your stack? The other reason is that one of the value parameters passed to FMTINIT' is the location in the caller's code segment of the Condition Code test that immediately follows the call to TFORM'; FMTINIT' tries to exit directly to this location in the event of an end-of-file or some other error--which it can't do if its above stack marker is pointing to the wrong code segment!

So, in order to mesh properly with the other unhooked Formatter calls, your FMTINIT' hook must somehow get completely out of the way once it's finished doing its thing, so that the real FMTINIT' may interface directly with the subprogram that called the hook, not the hook itself. What I'm talking about here is somewhat analogous in routine office life to getting a telephone call destined for another employee and transferring the call to that employee's extension, instead of relaying messages back and forth.

Sound impossible? I'm happy to report that it's quite do-able; but your hook must call FMTINIT' with the EXIT instruction instead of PCAL, because you have to get into FMTINIT' without generating another stack marker in transit (over whose return vector you would have absolutely no control) or causing anything to be deleted from the stack that is expected to be there for list element transfers.

Basically, your hook procedure's job is to intercept an LU number and use it to obtain from the 77 FLUT an MPE file number, which is then passed as a negative integer to FMTINIT', thus bypassing the 66 FLUT. From inside your hook, value parameter UNIT is located at Q-8 and specifies an LU number if UNIT > 0 and IOTYPE(9:1) = 0. Simply pass the LU number by reference to integer procedure F77_FNUM--which works the same way as its senior counterpart, except that it can dynamically allocate a FLUT entry--negate the returned MPE file number and store it at Q-8, overwriting the original LU number in the parameter list.

NOW the real fun begins! You need a support procedure--we'll call it XFERCALL--with two parameters: the number of the code segment containing the real FMTINIT'; and its entry point address, with the mapping flag in bit 1. The segment number is determined at either load time or process-creation time, depending upon whether the mapping is "physical" or "logical", but this and the mapping flag may be extracted from the *plabel* returned by LOADPROC. The entry point address was fixed as of your last MPE V upgrade; the Segmenter command "-LISTSL ENTRY,FMTINIT'" can tell you exactly what it is.

After saving its passed parameters locally, XFERCALL builds a new stack marker as follows:

Q'+1 Old X register from the above stack marker, i.e. from Q-3.

Q'+2 Entry point address and mapping flag as passed.

Q'+3 Status bits from the above stack marker, i.e. from bits (0:8) of Q-1, merged with the 8-bit segment number as passed.

Q'+4 Delta-Q = 4.

Your goal is to have the Q and S registers both pointing at what is now Q'+0 when the EXIT is consummated; this is where they were pointing immediately following the PCAL to the hook procedure. Since the SDEC, being held in the lower 8 bits of the EXIT instruction, cannot be larger than 255, the Q of the stack marker through which you propose to EXIT cannot be more than 259 words out from your target top-of-stack, so the best place for the new stack marker is adjacent to the hook's above stack marker, since nothing else between there and XFERCALL's local stuff needs preserving at this point; that way, it doesn't matter how much local stack your hook procedure uses. (Even if it doesn't use any at all, you've already copied XFERCALL's passed parameters out of harm's way, and its parameter list itself gives you just the amount of slack you need.)

The last step is to set the Q register to point to Q'+4 and ASSEMBLE an "EXIT 0". The hook's above stack marker and parameter list thus become the above stack marker and parameter list for the real FMTINIT'--just as though the application program had called it directly!

The other FORTRAN 66 externals--FNUM, FSET, UNITCONTROL and FINAUX'--may be hooked to their FORTRAN 77 counterparts by conventional intermediation.

IX. The "Green Area" and the FPMAP: How to Have Both

In the aforementioned INTEREX '90 paper on hooking, the problem of saving the *plabel* of a time-critical intrinsic between calls to a hook procedure raised the issue of global storage. Eugene's discussion of the possible solutions included the technique, developed by Robelle Consulting's Robert M. Green, of expanding the DB-Qi stack region by adroitly editing the PROG file after preparation. For our purposes, I will refer to this inserted global storage as the *Green area*.

What makes the Green area an especially good place for SL procedures to save stuff between calls is that the program, being logically unaware of its existence, will never use it, so there is no possibility of collision--at least, not with the program nor with any supported subsystems. Insertion of the Green area can be done with any existing program without re-PREPping it; but it does require a new copy of the program file with several sections relocated--the segment set, the entry point list, the external list and the FPMAP (if present); their starting record numbers held in record 0 (SAS, SAE, SAX, SAPMAP) must be increm-

ented by the number of 128-word records appended to the global area. Another record number that may need incrementing is for "symbolic items" (SASI); I've found no programs in our shop that use this field, but you may want to check it out.

Eugene's counsel was that any FPMAP be done away with by setting SAPMAP to $0\frac{17}{}$ because of all the internal record pointers that would have to be changed. The September '84 edition of the Tables manual, at any rate, does not describe these pointers in detail. But as we have seen, there are cases, as in Section V, where the FPMAP is essential; it would be nice if we could also enjoy the benefit of a Green area.

I've done some exploring with DISKED5 and found that retention of the FPMAP really isn't all that difficult--especially considering that Green area construction is already labor-intensive enough to warrant encoding the operations in a stand-alone utility. Since all program segments are prepared together, there is only one multiple-segment FPMAP serving the entire program, and it occupies contiguous records at the tail end of the program file data.

Word 1 of record 0 gives the number of code segments in the program (NS), and this is exactly the number of FPMAP record pointers that need incrementing by the same amount as all the others. They all occupy consecutive double-words constituting the "SEGMENT PMAP POINTERS" section (SPP), which immediately follows the "PMAP TYPE TABLE" (PTT) at the very beginning of the FPMAP. To allow for future expansion, the PTT length (PTTL) is given in word 0 of the PTT. Thus, the SPP is located at record SAPMAP, offset PTTL, and its length is 2*NS words (maximum 510). Each pointer in the SPP is in the standard 32-bit format found in USLs--a record number in the upper 25 bits and a word offset in the lower 7--and points to a Type 0 FPMAP entry, which is followed by the entry point entries (Types 1 and 2) for that segment in ascending STT index order, followed either by a null word or by the Type 0 entry for the next segment in ascending LOG# order. Only the record number field in each pointer needs adjustment.

There are no other file-dependent pointers that you need worry about. Other than flag bits for hidden entry points, FPMAP entries contain nothing that isn't documented in the Segmenter manual or returned by system intrinsics.

A Zero-Defect Method of Locating the Green Area

How does the Green area say "Here I am!" to a hook procedure that's looking for it? Eugene suggested that you logically begin the Green area on an even 128-word boundary and initialize the first few words to some unique bit pattern that is highly unlikely to be coincidentally duplicated elsewhere in DB-Qi at such a boundary. This bit pattern would have to be hard-coded in every hook procedure that uses the Green area so that they would know what to look for at 128-word intervals beginning at DB+0.

Now, the operative phrase here is "highly unlikely", which implies an anticipated failure rate ever so slightly larger than 0. The challenge of finding a 100%-reliable method using existing structure without conflict was in the back of my mind one day as I was idly scanning through the Compiler Library manual; my roving eye fell upon the detailed layout of the FORTRAN 66 FLUT, and--would you believe it?--I found the answer right there!

As I mentioned in Section VIII, the Segmenter and the Loader support the 66 FLUT on MPE V systems. In the program file, word 12 of record 0 (SAFLUT) gives the DB-relative address of the FLUT, or defaults to -1.

According to both the Tables and Compiler Library manuals, the FLUT terminator word is characterized only by a 255 in the upper byte; the lower byte is always initialized to 255 but is not used for anything, and testing has confirmed that it isn't. You can put anything you want in there without interfering with the Formatter.

What does all this have to do with locating the Green area?

Well, if we adopt the convention that the Green area always start on an even 128-word boundary, then its DB-relative address is evenly divisible by 128--and the quotient fits very neatly in 8 bits! Since there is an initialized but unused byte at the tail end of the FLUT, what a wonderful opportunity to chain therefrom to the Green area! The starting record number of the global area is given in record 0, word 3 (SAG); the compressed address of the Green area works out to be <Green area record #> - SAG. The FLUT itself is located starting at record SAFLUT(0:9)+SAG, offset SAFLUT(9:7).

Yeah, but what if it's a COBOL program and doesn't have any FLUT?

No problem--if we further stipulate that the Green area always begin with a word containing 255 in the upper byte, and <Green area record #> - SAG in the lower, then if SAFLUT is defaulted, we need only set it to 128 * (<Green area record #> - SAG). If, at process-creation time, the Loader finds anything greater than -1 in SAFLUT, then that number will appear on the process' stack at DB-1, no matter what (in DB-Qi) it's really pointing to. This way, the Green area can be located at run time by the same method whether SAFLUT is pointing to a FLUT or directly to the Green area.

So: At run time, to find out if there is a Green area and where it is, first check the DL register; if it is 0, there is no Green area, otherwise read DB-1; if it contains -1, there is no Green area, otherwise scan the FLUT for the terminator word; if the lower byte contains 255, there is no Green area, otherwise that number multiplied by 128 gives a DB-relative address that should contain an exact copy of the FLUT terminator word; if it does--

bingo!

This design allows for dynamic relocation and subsequent expansion of the FLUT (which could initially be found empty). As long as you always leave the terminator word content as you find it, you will not corrupt the Green area indicator/pointer.

But what if the compressed address of the Green area happens to be 255? Wouldn't it appear that it didn't exist?

Let's see: 255 * 128 = 32640; the largest possible data segment of any kind is 32764 words, and the maximum allowable MAXDATA is usually less than that; that leaves not more than 124 words for other objects in the stack besides a global area. As many as 128 words of heap space must minimally be allocated for the FLUT pointer. How much room would that leave for the PCBX (or anything else)?

Green Area Builder Design Points

To determine how large your Green area is permitted to be, close attention should be paid to the following record 0 fields: initial stack size (ISS); initial heap size (IDLS); and MAXDATA (MAXD). The following SYSGLOB[18] cells are also relevant:

%107	Maximum allowable MAXDATA.
%110	Default MAXDATA.

When a program is prepared without specifying a MAXDATA, MAXD is set to -1 to indicate that the default value in SYSDB+%110 is to be used. If MAXD contains a positive word count, it may be increased as needed.

A Suggested Application

A custom procedure designed for the EDITOR's /PROCEDURE command[19] has a total of 30 words of global storage available. Is that enough? Would you like more? Would you like to have complete control over what is in your global area at any given time?

The EDITOR program as supplied has a MAXDATA of only 10,000 words, so there is plenty of potential space for as large a Green area as you would ever want. The only problem to be addressed is that the EDITOR obviously has no way of re-initializing any part of the Green area after the specified *rangelist* has run its course.

The solution lies in the fact that your custom procedure is called by the internal procedure that processes the *rangelist* and which only exits to the main program after the final iteration. Using the material in Section III, you can hook this exit to call any SL procedure charged with the task of cleaning up. Since the Green area can be initialized at the time it is built, I can't

think of a better place for the exit trap stack.

One word in the Green area should be used for a flag to indicate whether EXITRAP has been called for the current *rangelist*.

X. The Rest of the "Rest of the Story"

Here, in full, is the supplemental material that I intended to hand out in New Orleans.

Who's Using That Code Segment? IF Anyone, and IF It Exists

In Application Example 3 in *S-II*, I told you the minimum you need to know in order to write a utility program that reports SL code segment usage. Unfortunately, I neglected to tell you how to determine up front whether it is needful, in the case of any given SL segment, to impose a lock on such a critical system bottleneck as the Loader Segment Table (LST) and go to the fair amount of trouble to read it.

In the interests of system performance, your program should not read the LST or lock any SIRs, pass GO or collect $200, without first making sure that the specified code segment exists in the specified SL.

Prepared Object Code file structures are described in the Tables manual, Chapter 10; the September '84 edition is not up to date with respect to the system library.

In general, each code segment in the SL has a single entry in a Reference Table (REFTAB), and the table-relative position of this entry defines its LOG#, which does not appear in the entry itself. However, the records containing the table entries are not contiguous; for this reason, they are listed in a block of record numbers (RTIDX) beginning at word 0 of record 1.

A REFTAB entry contains, among other goodies, a bit map indicating which code segments in the SL are referenced by the current segment. How large is this bit map? If you guessed 32 words in the system library, and 16 words in a user SL, you're on the money! Consequently:

* The REFTAB entry size is 64 words in the system library, and 32 words in a user SL.

* REFTAB entries are packed 2 to a record in the system library, and 4 to a record in a user SL.

* The maximum size of the RTIDX is 255 words in the system library, and 64 words in a user SL.

A REFTAB entry for an allocated LOG# not currently assigned is

indicated in entry word 3 if bit 0 ("SEGMENT DELETED") is set to 1.

Remember that the highest legal LOG# is %375 (253) in a user SL, and %775 (509) in the system library, and is not necessarily the highest LOG# currently allocated. The current number of REFTAB entries (NRT) is in word 9 of record 0, and your LOG# must first be compared to this number--not the number of segments in word 4; if the LOG# $\geq$ NRT, it is no good.

If we let *P* = the number of REFTAB entries per record (either 2 or 4), then the actual size of the RTIDX is:

$$(NRT+P-1)/P$$

The record number containing the REFTAB entry for LOG# *N* is in:

$$RTIDX(N/P)$$

The word offset of the entry within the record containing it is:

$$128/P*(N \text{ modulo } P)$$

If the "SEGMENT DELETED" bit is on, your LOG# is no good.

If you allow a code segment to be specified by name, you must serially read the REFTAB until you find a "good" entry containing the matching segment name in words 8-15, left-justified with blank padding. The REFTAB records must be read in the order given in the RTIDX. Before commencing the search, you must initialize a counter on your stack to -1, and increment it by 1 and compare it to NRT before each inspection of an entry; the counter value at the end of a successful search is the LOG# of your code segment. (Remember that a "good" entry has a non-negative value in word 3, but all entries must be counted.)

Once you have determined that your LOG# or segment name is valid and existent, you need to make sure that it's loaded. You can do this by using your target SL's file disc address to find its Type 1 entry in the LST, and scanning the SEGLIST for the LOG#.

Not so fast, though! Before you even consider going into the LST, you should first take a closer look at the REFTAB entry. Word 0 contains a bit called SLSEGFLAG, indicating whether the STT is in the old pre-MPE V format or in the new format for logical/physical mapping[20] When a segment is "born", i.e. added to the SL, the STT is initialized the old way, and SLSEGFLAG is set off; as soon as it is "christened", i.e. loaded for the first time, it is upgraded to MPE V standard, and SLSEGFLAG is set on for keeps. So, if you find it off, your segment is not, and has never been, loaded; if it is on, your next step is to read your SL's file label and check the FLLOAD flag; if this bit is 0, it means that none of the segments are loaded--in which case, obviously your target segment isn't, either.

The Trouble With Helpful Harry

In *S-II*, I recommended Boeing Technology's contributed procedure SEG'READ, for reading data segments, as a building block for the custom trace-back procedure. When used in conjunction with its companion piece SEG'ERROR, it provides the level and quality of error-checking and "looking before leaping" that are essential to have in any situation involving SIRs. However, if you actually tried calling it from your TRACEBACK, then even if (or perhaps, because) you followed my instructions to the letter, I have reason to suspect that things didn't go very well at first--do I guess rightly that your test program bombed on an "ILLEGAL CAPABILITY" error, and took the system down with it because the LST SIR was locked at the time?

I apologize for forgetting to warn you about this, but here's what happened:

SEG'READ, SEG'WRITE and some other contributed procedures that use privilege do not count on you to be running in Privileged Mode when you call them, so they call GETPRIVMODE for your convenience; but this is not very convenient at all unless the program has PM capability, and the whole point of working from "OPTION PRIVILEGED" is to make TRACEBACK available to all programs.

The best solution, if you have the source code and the SPL compiler, is to produce a new copy of SEG'READ with all calls to GETPRIVMODE and GETUSERMODE taken out and the keywords "PRIVILEGED" and "UNCALLABLE" added to the OPTION list; a slight change of entry name may be desirable for parallel compatibility with existing PM programs that call the original SEG'READ.

One More Note on *Plabels*

COBOL II allows you to call a procedure given the *plabel* even more easily than does FORTRAN. If *identifier-1* in the CALL statement is numeric, it is assumed to contain a *plabel*, and no LOADPROC is done--and no "gateway" procedure is necessary.

But if you've read the COBOL II manual as it were a love letter, you already know this.

Conclusion

John Dunlop, an MPE user in Britain, in his letter published in the September 1991 *Interact*: "I just received the June 1991 *Interact*...and I read through it with considerable dismay. The entire magazine was taken up with UNIX, HP-UX, POSIX, ALLBASE, and ORACLE--none of which I have any interest in. ...I am sorry to see *Interact* become such a market-oriented magazine as opposed to the technical and informational source it has been."

In the course of my fourteen-plus years at United Electric, I have seen my company grow in the sense of maturing and evolving into the 90's; it has progressed from being one of many small businesses adhering to traditional manufacturing and managerial methods to being a Shingo prize-winner. Consequently, our MIS department has kept pace much more with the changing needs and realities of our company's business, and ways of doing business, than with advancing HP software and hardware technologies, and this has made it necessary for me to find ways to actualize, under versions V-Delta-4 and earlier, some software features similar to those becoming standard with Release 30 and later, using only the limited software and documentation resources available to me. So, I have experienced Mr. Dunlop's kind of frustration at finding little new and interesting published about the system I have been working with for nearly a decade, and at finding no answers to lingering questions while having very limited time to pursue them--not that other systems currently in use, just coming out, or on the horizon shouldn't also be discussed with enthusiasm.

This is probably my final paper on ye Antient & Honourable MPE V or its subsystems. May INTEREX and my readers forgive its length, but time is pressing to share as much of my Classic MPE knowledge and expertise as possible before it goes the way of alchemy, and I entertain the hope that other shops still happily using this mature and reliable system--if only to maintain some stability in the face of ever-more-rapid technological change--will derive benefit therefrom, and never again be stonewalled by such defeatist statements as "such-and-such can only be done in SPL".

It has never been my primary intent to add balance to the mix of writings on HP software--but if I have, so much the better.

Finally, I would like to thank Eugene Volokh for his expert advice--you will never know how much trouble he may have saved me from causing!

* * * * *

To all those who dare greatly for the sake of knowledge

NOTES

1. Reprinted in Thoughts and Discourses on HP3000 Software (VESOFT, Inc., Los Angeles, CA), 3rd edition.

2. 1992 INTEREX Proceedings, New Orleans, LA; Paper #3051.

3. Some system intrinsics are documented as "O-P"; this does not mean "OPTION PRIVILEGED". In the case of GETPRIVMODE, it means that the calling process must have PM capability; in the case of SWITCHDB, it means "OPTION UNCALLABLE", i.e. not callable in User Mode.

4. This approach is rendered less than totally reliable by the fact that on version V-Delta-4 and earlier, PROCINFO does not indicate whether the program file is permanent or temporary; I made much of this in S-II. Check the most current documentation for your MPE V platform.

5. MANMAN is a registered trademark of:

   ASK Computer Systems Inc.
   2440 W. El Camino Real
   Mountain View, CA 94039

6. Media files are usually opened globally by TurboIMAGE.

7. We use the CSL procedure GENERIC.

8. FORTRAN 66 label parameters are recognized by the Segmenter. The external reference entry has a subentry for a single parameter of type LABEL (FORTRAN). Unless coded in the same language, ISCAL's formal checking level must be no higher than 1.

9. The same method required to exit from a CONTROL-Y trap procedure armed via XCONTRAP; see the Intrinsics manual, Chapter 5.

10. For full details on the Formatter procedures and calling sequences, see the Compiler Library manual, Chapter 1.

11. Does anybody remember the old Bert and I recordings?

12. My own nomenclature from S-II. HP calls it a "logical" segment number--the same term they use for a code segment in Prepared Object Code.

13. FINDPMAPNAME returns information only for revealed entry points.

14. FORTRAN Logical Unit Table.

15. 1993 INTEREX Proceedings, San Francisco, CA; Paper #5035.

16. 1990 INTEREX Proceedings, Boston, MA; Paper #3141.

17. In S-II, the TRACEBACK logic could have checked the state of SAPMAP in Step 1, and FCLOSEd the program file right then and there if SAPMAP were found to be 0.

18. SYStem GLOBal Area. The undocumented non-privileged integer procedure SYSGLOB may be used to read it; see Eugene Volokh, Thoughts and Discourses on HP3000 Software, 4th edition, page 153.

19. See EDIT/3000 Reference Manual, Chapter 4.

20. See David N. Holinstat, "Architectural Changes for MPE V"; 1983 HPIUG Proceedings, Edinburgh, U.K.; Paper #70.

5043

How to Win at Your Next EDP Audit!

Tom Harris
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road, Suite 106
Palo Alto, CA
(415) 493-4122

As data center manager, you often walk a precarious tightrope, successfully meeting corporate objectives while directing the day to day processing activities of a busy data center. And during your daily review of the data center, some concerns may go unnoticed. These are the areas most often uncovered during an EDP audit, and they can result in the imposition of unnecessarily stringent requirements on the data center.

This paper attempts to turn a tough assignment — facing an EDP audit — into a routine exercise by focusing on those concerns most often overlooked by the data center manager. By understanding what the EDP auditor looks for during a review, you will have the knowledge to prepare your data center to pass an audit with flying colors. This paper will also arm you with a checklist for improving and automating your operations, and help you to successfully anticipate even the toughest EDP audit.

PROBLEMS ADDRESSED

During a review of the data center, the auditor is chiefly concerned with the efficiency and security of DP operations in the following areas: standards and procedures, operational work flow and controls, scheduling, data security, change control, equipment utilization and efficiency, disaster planning and recovery, and environment.

If the data center manager and staff understand what the auditor is looking for and what information is needed by the auditor, the review can proceed more smoothly, and the results can be more beneficial to the organization.

STANDARDS AND PROCEDURES

Standards and procedures that should be in place and enforced include:

- Ensuring proper timing in running programs, inserting changes into programs, and using the correct data for programs
- Protecting the data and programs from accidental or intentional destruction
- Ensuring that the data processed is complete and accurate
- Specifying methods of physically moving input and output
- Scheduling work and getting work rerun in the event of an error or disaster

- Specifying procedures for controlling data, programs, and the flow of work
- Keeping records of work performed
- Determining and recording sufficient resources for the work
- Performing maintenance and general housekeeping associated with the operation of the computer center

The EDP auditor may also want to verify that formal standards exist for systems development and maintenance, program and system change control, library operations, computer operations, and documentation.

OPERATIONAL WORK FLOW AND CONTROLS

The auditor will investigate specific items in this area, including whether:

- Input data from other departments is complete and entered on time
- The data center keeps job accounting information
- Job accounting information is evaluated and used by management

Error control procedures should also be reviewed. Specific questions asked include:

- Is anyone notified in case of a production processing error?
- Are errors documented?
- Are error statistics accumulated or ignored?
- Are errors followed up on so that they do not recur?

The auditor will also confirm that downtime is reported and statistics compiled. A log of late reports and jobs should be maintained.

There should be a formal communications channel between data center operations and other departments; operational tips and other advice should be passed to all operators.

All problems encountered at the computer, as well as any action taken to prevent their recurrence, must be documented. Operators must also receive feedback on reported problems. The auditor will verify that headers and $STDLIST information is used and checked.

Next, the auditor scrutinizes output report distribution and disposal and determines whether:

- All reports have been distributed to the proper user
- Procedures have been established to control the distribution of sensitive output
- Procedures exist for disposing of confidential reports when they are no longer required

Finally, the auditor will want to ensure that jobstream run instructions are kept up to date.

SCHEDULING

Efficient and effective scheduling is extremely important in providing a high level of reliability and predictability to DP operations.

The auditor will determine whether:

- Daily processing activities are scheduled and a daily contingency schedule is maintained. A strict schedule for nightly batch runs should also be established and adhered to.
- Actual run times are recorded for batch programs.
- This data is used to calculate expected run times to ensure that runs have not been terminated abnormally.
- Unscheduled runs are supported by a work request or other written authorization. Schedule deviations should be documented and followed up on by a supervisor.
- User-submitted jobs are recorded to allow forecasting of future schedules, resource requirements and special processing considerations for online systems.
- All jobs are submitted through or controlled by data center operations. All output should be routed by operations to the appropriate destination or picked up by the user.
- Standards cover the type, quality and quantity of forms kept on hand.

DATA SECURITY

Data base information should be protected from unauthorized access or loss. Employees must be instructed about their responsibilities concerning confidential information. Management must periodically review and update controls and security provision relating to data. Live production programs should be physically separated from development programs. The staff should be prohibited from running test programs against live files, and operations personnel should be denied access to sensitive data files.

To maintain security, operators should be prohibited from renaming or transferring programs without supervisory approval. Internal labels must be used for all data and program files.

Passwords and lockwords should be used to protect accounts, users, and data files. Passwords, lockwords, dates, and constants should be introduced at run time, eliminating the need to hard-code sensitive data into jobstreams.

Access violations must be logged and reported to the security manager. An automatic log-off feature prevents unattended terminals from posing security

threats. The auditor should examine the area above the suspended ceiling in the computer room to confirm that it is accessible only from that room.

The auditor investigates blank check stock and other negotiables to determine whether they are issued on a run-schedule basis, kept in a secure area when unattended, controlled by access forms, and periodically inventoried.

Every site's security needs differ according to hardware, business focus, personnel, system function, work schedules, work environment and numerous other variables. Having stated this, there remain several constants which can help you determine your security needs. Take a moment to answer the following questions:

- Are data processing employees instructed as to their responsibilities concerning confidential information?
- Are live production programs physically separated from development programs?
- Are program library changes approved and accounted for?
- Are operators prohibited from renaming or transcribing programs without prior supervisor approval?
- Are internal labels used for all data and program files. Is an operations log maintained?
- Is the area above the suspended ceiling in the computer room accessible only from that area?
- Are blank checks and other negotiables issued on a run schedule basis only?
- Do you ever have the same password for more than thirty days?
- Is sensitive data endangered by sessions that remain on unattended terminals?
- Has your auditing firm asked for stricter reporting standards?
- Is management spending too much time implementing password changes?
- When a person who has access to sensitive information leaves your organization do you globally change passwords?
- Can users log on to any terminal?
- Are your ports being tied up by people who fail to log off?
- Can users log on to terminal from remote ports?
- Can users log on from remote ports at any time of day?
- Are additional passwords needed to log on remotely?
- Can users circumvent existing procedures to run jobs during off hours?
- Would your security be enhanced if passwords were not embedded into jobstreams?
- Do users have access to sensitive jobstreams?

CHANGE CONTROL

Change control procedures for computer programs should be established and followed. The intent of these controls is to prevent unauthorized, inaccurate, and

unreliable program changes from being incorporated into the live production environment. Both scheduled and emergency changes must be appropriately controlled to maintain the ongoing integrity of software.

The auditor will check to see that the following techniques are in place to ensure that proper controls are being maintained over your program changes:

- Develop and adhere to formally approved written standards for all program changes
- Define and enforce procedures detailing who can initiate and who can authorize program change requests
- Describe and track the nature and reasons for proposed changes
- Enforce testing and acceptance procedures for all program changes including emergency changes
- Test all program changes under normal operating conditions
- Involve users in preparing test data and reviewing test results
- Investigate and correct all errors before transferring code to production
- Certify that all test results demonstrate adequate protection from fraud, waste, and misuse of the program
- Document all program changes and update appropriate documentation as changes are made
- Log all completed changes as well as those changes in progress
- Utilize a formal system to report all changes to users and project managers
- Enforce a checkout-checkin procedure that prevents a file from being simultaneously modified by more than one programmer
- Develop procedures to analyze whether other systems are affected by new program modification
- Retain and secure original source code until changes have been processed, tested and updated
- Limit the frequency of program changes, except for emergency cases
- Notify both the user and EDP project manager when emergency changes are made

EQUIPMENT UTILIZATION AND EFFICIENCY

Once it has been determined that the entire DP department is following a properly implemented set of standards and procedures, the auditor may wish to review equipment utilization.

The auditor will determine how much machine time is spent on reruns, whether reruns are analyzed, and whether certain jobs are especially susceptible to reruns. The auditor will also review programs or jobs for insufficient file design or utilization. Another area to check is the full multiprogramming capability of the system for batch production. The auditor will determine whether multiple jobstreams run concurrently and whether CPU-bound and I/O-bound jobs are

mixed to maximize overall throughput. The auditor then reviews whether many jobs can be restarted without rerunning the entire job.

PERSONNEL USE AND EFFICIENCY

Review of personnel practices can be a sensitive issue. Key areas of interest include:

- Do operations personnel require extensive training and experience to be effective in processing daily production work? Is extensive knowledge of each application run necessary?
- Is there a system to schedule and monitor regular daily processing? Is the system effective? Does it operate without excessive human involvement? Or do operators spend a large part of their time tracking jobs in execution, replying to program messages, and changing job priorities? Must operators modify jobstreams at run times?
- Are all necessary tapes, forms, and other resources available when needed?
- Is there excessive turnover? Does daily production depend on specific individuals?
- Is the operations department treated as less important than the rest of data processing?

DISASTER PLANNING AND RECOVERY

This catchall category includes everything from proper insurance planning to physical security procedures. The auditor will want to determine whether the emergency plan is adequate in relation to the risk. This plan should be kept current and distributed only on a need-to-know basis.

The plan for off-site storage of files and documentation should specify:

- The conditions for use of off-site processing
- Processing priority for applications
- Resource requirements
- Job scheduling
- Run documentation
- Required tapes, forms, and supplies

Formal procedures for hardware backup should also be instituted.

ENVIRONMENT

The auditor may wish to review the work space to ensure that it is adequate for the number of employees. The environment should be neat, and supplies should be easy to locate.

Auxiliary items located outside the computer room, such as bursters and decollators, should be accessible for the flow of work in the department. Tapes, disks, and other storage media should be stored in a closed, fire-protected, limited-access area.

RECOMMENDED COURSE OF ACTION

By understanding what the EDP auditor looks for during a review, you can now prepare your data center to pass its next audit. The checklists provided in this paper will help you to successfully anticipate even the toughest EDP audit. The data center manager should be aware that the following more general advice can also greatly enhance the data center review:

- Provide the auditor with as much information as possible
- Implement a software system that leaves clearly defined audit trails
- Keep accurate records
- Maintain formal written standards and procedures
- Implement an effective data security system and maintain an emergency plan
- Follow the auditor's recommendations and procedures in preparing for future audits to ensure efficient and cost-effective operations

Its hard to remember your objective when your up to your @#$ in Auditors

You're at your desk and the phone rings, the voice on the other end repeats these words "Your Being Audited", three words that brings sweat to the brow of Systems Managers, Development Personnel and Mangers of Information Systems.

What this paper will present is an overview of "How to keep daily activities going and supply your auditors with the documentation to conduct a successful EDP audit of your center". Being prepared is the key to a successful audit, the topics of discussion Physical Security - do you have proper security controls on your building, access to your computer facility, tracking of all entry. Login Security - Passwords are they changed on a regular basis, and do you have a log of when, and who changed the passwords. The information systems today are the life source of most organizations, protecting the computers and the data is standard operating procedure. Access control to live data - who has it and why, can the data be changed and not controlled. Do you have in place a mechanism to move test programs (files) to production and are they logged, and verified before moving to production.

Do you have in place control procedures that support your Systems,Operations and Development Staffs, and the key item that auditors will ask for is your "Procedure Manual" can you find it? and better yet, is it updated. Located in your set(s) of manuals would be items such as Management of Networks, Monitoring of System Performance, Hardware and Software contact list, Procedures to set up new users,groups and accounts along with signoffs to obtain OP, PM capabilities. Recovery Procedures for startup, shutdown and failures of hardware.

Tools - The auditors want specific reports - can you produce in a timely manner and what is not being done while your staff is fulfilling the auditors every request.

Do you have a security controls in place, if not look at the many vendors available today that can offer a solution. Is your change control system in place, and how does it help/hinder the work flow of your organization. The many tools available today will be discussed - ie OCS, MPEX, ALDON, and Engarde what their roles are at the university, and how they help keep the work going as fill audit request daily.

The intended audience is anyone who may come into contact with either internal or external auditors, dealing with various questions directed to Systems,Operations,Development, and Management who responds to audit points.

J. Michael Lam -

I have been with James Madison University for the past 18 years, Manager of Systems/Operations and Security of all the university owned computing systems. Managing multiple Hewlett Packard MPE-IX systems, Hewlett Packard HP-UX systems and Digital VAX VMS systems supporting over 11,500 students and 2000 full time faculty and staff. Degree in Computer Science and business management, installed the first HP 3000 Series II in 1976. I have attended HP Interex conferences since 1983, and more recently attended conferences on the Control, Audit and Security of Information systems.

Paper 5045

SOME USEFUL HP 3000 SYSTEM MANAGEMENT TECHNIQUES

by Gilles Schipper

The manager of today's HP 3000 computer system faces contradictory pressures. On the one hand, advancing technology has brought about increased reliability and much faster processing speeds. This in turn has greatly simplified and quickened day-to-day tasks such as backup, communications, and batch processing.

On the other hand, this superior technology has led to significantly increased processing throughput and increased functionality. Batch jobs that previously took hours to finish are now measured in minutes. A couple of years ago, who would have given a thought to using the HP 3000 as a Netware server? Now, not only is it possible but Netware/3000 has excellent price/performance and makes sense for many HP 3000 sites.

One thing, however, has remained constant over the years. The right mixture of common sense and know-how can streamline and simplify the task of managing and operating your HP 3000 system.

Some of the following techniques and tools that can make your life easier are Classic or Spectrum specific, and will be so indicated where applicable.

For those unfamiliar with the terms "Classic" and "Spectrum," the Classic HP 3000 is that family of older HP 3000 computer models that runs the operating system known as MPE VE or prior. This includes the following HP 3000 models: Series 30, 33, 37, 39, 42, 44, 48, 52, 58, 64, 68, 70, Micro3000, Micro3000 GX, Micro3000 LX.

The Spectrum family of HP 3000 computer models runs the operating system known as MPE XL or MPE/iX. These include the following HP 3000 models: 917, 920, 922, 925, 927, 930, 932, 935, 937, 947, 948, 950, 955, 957, 958, 960, 967, 977, 980, 987, 990, 992. Most of these models may also carry an LX, RX, or SX designation.

1. In your full backup, use SYSDUMP (Classic) or SYSGEN (Spectrum), instead of STORE. This ensures that you have a bootable tape (cold load tape for Classic; SLT tape for Spectrum) that's never older than your last full backup. As a Classic user, you will appreciate this the first time you are forced into an unplanned reload. A reload from an old SYSDUMP tape brings you an old directory structure including stale accounts, groups, and users, together with ancient passwords.

Spectrum users will not suffer the same consequences, since the system directory is not included with the SLT. However, as with Classic systems, you may still suffer the consequences of non-current system files and/or IO configuration when forced to install from a stale SLT.

Spectrum users may not know that as of MPE XL 3.1, it's possible to combine the SLT with user and system files on a single backup set. It's so much more

convenient, with the added benefit of ensuring the availability of a reasonably current SLT.

2. While on the subject of backups, consider using the following file set on your full backup file set:

```
@.@.SYS,@.@.@-@.@.SYS
```

(Add ;DIRECTORY, for Spectrum)
or, if on Spectrum release MPE/iX 4.5 or beyond,

```
/SYS/,/ - /SYS/;DIRECTORY
```

This ensures that the SYS account is stored first on your backup tapes. The reason you want the `-@.@.SYS` (or, `- /SYS/`) is so that you do not clutter up your output listing with the misleading message "UNABLE TO STORE filename" for each file in the SYS account. This would only discourage you from carefully perusing your output listing for files not stored.

You can extend the principle of storing accounts in order of importance by constructing an appropriate file set with corresponding exclusion file sets to avoid the aforementioned misleading warning messages. However, keep in mind you can include at most nine global file set exclusions on Spectrum systems, and only one on Classic systems (Platform 3P, the newly released latest version of MPE VE allows you nine). For example,

```
@.@.SYS,@.@.PROD,@.@.@-@.@.SYS-@.@.PROD
```

3. If possible, try to avoid using your physical console (logical device number 20) for anything other than emergencies -- for several good reasons: (a) As long as I can remember, "break"ing and "abort"ing a program at the console can cause the system to hang. This is true for both Classic and Spectrum versions. Various patches have temporarily fixed this pesky problem, but it reappeared after upgrading to the next MPE version. (b) Regular use of the console invites the risk of system hangs due to incomplete typing of a command. This problem is more severe for Classic systems. You don't really want to try this, but you can test out this "feature" by typing a few characters at the console, then walking away. If you're looking for a little excitement, maybe you can place a bet with a colleague on how long it will take for the system to hang. (c) Important console messages could scroll off the console screen unnoticed if interspersed with subsystem output (see 4 below). (d) Console messages can interfere with your activities while you are working at the console.

I recommend keeping the physical console logged off. Contrary to popular belief, MPE does not require an active session at the console in order to conduct its operating system activities properly. (See below for recommendations regarding alternate "logical" consoles.)

4. If possible, use a "logical" console different from LDEV 20. You may then acquire all console command capabilities by getting a copy of ALLOWME (from the Interex contributed library). Before doing so, however, be sure you log on to that logical console with ;PRI=BS. (You may need to have the system manager

assign you that capability at the account and user level.)

If you do that, you can always be king of the castle and gain access to any critical system resource that you would otherwise be unable to access even by simply logging on to investigate.

5. Acquire hard-copy console capability. This can be satisfied simply by attaching an inexpensive serial printer to your console's auxiliary serial port. In fact, the newer HP700/96 family of terminals also includes a parallel printer port.

This capability can save your bacon. Consider, for example, a situation in which a very important message escapes your careful attempt to jot it down in detail because of the speed at which it rolls off the console screen. Even with a utility such as PSCREEN, you may not be able to have a hard copy of the message because (a) there is not enough terminal memory to retain the message, or (b) your system is not yet up and running MPE. Also, the sheer volume of output you should be capturing (for example, in the rare, but critical, case of storing user files to tape with the stand-alone DUS utility) mandates a hard-copy capability.

You could even consider routinely enabling "log bottom" on your console terminal to give you permanent console output hard-copy logs, rather than using the hard-copy feature on an exception basis.

6. Use BULDACCT to assist in volume management and disaster recovery. On Spectrum machines, this program is a fully supported utility included in PUB.SYS. On Classic machines, you can find this program in the TELESUP account, which typically means it is unsupported by HP, although it works just fine, thank you.

You should include a run of BULDACCT in your full backup job stream (and, yes, it really should be a job stream) prior to the actual "STORE" command. This will ensure these files will be backed up to your backup medium. Make sure the files created by this program (two files called BULDJOB1 and BULDJOB2 for Spectrum; three files called JOBACCTA, JOBACCTB, and JOBCUDC for Classic) are properly secured, since they contain embedded passwords for all accounts, users, and groups on your system.

You could use these files to recreate your system directory on any compatible HP 3000. You could also use them to reconstruct a dated directory structure resulting from a RELOAD (Classic) from an older cold load tape, or where an INSTALL (Spectrum) must be performed and a current directory file is unavailable (see 1).

7. If you are using Private Volumes (or Volume Management) to keep entire accounts off the MPEXL_SYSTEM_VOLUME_SET, you must not forget to issue the following command after setting up your private volume account structure:

```
:ALTACCT PVACCT01;FILES=0
```

This ensures that new groups created for the private volume account cannot accumulate permanent files on the MPEXL_SYSTEM_VOLUME_SET. The account manager (or system manager) will be required to :ALTGROUP

newgroup;HOMEVS=PVSET before that group can be used to hold permanent disk files.

Incidentally, if the task of created accounts and groups is exclusive to the system manager, no user other than SM requires CV or UV capability (contrary to popular belief).

8. Use a minimal SYSSTART file to gain control of your system, without which it is difficult to assume such control. Your SYSSTART.PUB.SYS file should look something like this:

```
JOBFENCE 14
STREAMS 10
STARTSESS 20;OPER/OPERPASS.SYSTEM;HIPRI;PRI=BS;NOWAIT
****
```

To complete the initialization of the balance of your production environment, create a job stream for that purpose. Then when you are satisfied that users can gain access to your system, simply stream the job STARTJ. (This job stream should probably use the aforementioned ALLOWME or GOD [by VESOFT, Inc.] in order to permit it to use console commands, such as JOBFENCE 7, for example.)

If you do that, you always retain complete control of your system --allowing you, if necessary, exclusive system access for specific trouble-shooting or maintenance requirements.

Without the initial JOBFENCE 14 setting, any standard user can gain access to your system by simply entering the appropriate :HELLO command, and there is little you can do about it.

If you use your SYSSTART file to set up your entire production environment automatically, you run the risk of allowing your production databases (or other files) to be accessed before you have had a chance to diagnose possible data corruption in the aftermath of a system failure.

9. While on the subject of SYSSTART, if you have had trouble with that file being ignored during system startup, it may be because the creator is other than "MANAGER." If that is the case, simply log on as MANAGER.SYS, and use your favorite editor (other than QUAD) to re-keep the file. (It seems that QUAD preserves the original file creator.)

10. Do VALIDATE your backups ON OCCASION. The frequency will vary according to several factors (backup medium, backup strategy, etc.). In general, validate your backups at least once quarterly, or as soon as possible following a major configuration change (new tape drive, disk drive, O/S revision, change to backup software, etc.)

On the Classic, use the program VALIDATE, which is usually found in the TELESUP account.

On the Spectrum, you will need to use the VSTORE command to validate tapes output with the "STORE" command. Additionally, you should use the program CHECKSLT (in the TELESUP or SYS account) to ensure a good SLT tape.

If you are using third-party backup software, use the validation routines that surely must accompany the backup software.

Never trust the validity of your backup medium. Even regular validation according to these recommended methods is not a 100-percent guarantee of valid backed-up data.

In addition to the above steps, I suggest restoring an oft-used account (complete with databases) to a different, newly created account (using the ;ACCOUNT= option of the RESTORE command) and running application programs against the data to test data integrity. This is the only way to guarantee data integrity of your stored files. This should not need to be done very often -- perhaps annually.

Although data validation can be time-consuming and cumbersome, it can almost certainly be done during normal daytime hours without a serious impact upon production.

This may well be the most important idea you bring back with you from this article.

11. While on the subject of validation: ALWAYS validate all tapes associated with your MPE O/S update BEFORE you actually perform the UPDATE.

Ever since Spectrum, HP -- for a reason which escapes me -- no longer sends instructions on how to validate your tape media together with UPDATE instructions and tape media. Perhaps it is because the necessary procedures are slightly more involved.

With Spectrum updates, you are usually sent three sets of tapes: (i) SLT tape set. (If on DAT, this is one tape. If 1600 BPI, this comprises four or five tapes depending on O/S revision. If 6250 BPI, it comprises two tapes.) (ii) FOS tape, until now (MPE/iX 4.5) in compatibility mode STORE format. (iii) SUBSYS tape containing optional HP software, also in compatibility mode STORE format.

You should also have a fourth tape set: the latest and greatest PowerPatch tape (if you subscribe to at least response center support). This last tape set is NOT sent automatically. You must (and should) ask for it just prior to your planned Update.

Use VALIDATE to validate your FOS, SUBSYS, and PowerPatch tapes. Prior to MPE/iX 4.5, the VALIDATE program was not included in either the SYS or TELESUP accounts. To get it, call your HP rep or Interex. They can get it for you or tell you how to get it. As of MPE/iX 4.5, you can find VALIDATE in PUBXL.TELESUP.

The SLT tape should be validated with the CHECKSLT.MPEXL.TELESUP program.

Also, your validation procedures are still incomplete if you do not restore @.@.SOFTREP;CREATE from your SUBSYS tape and print the single file contained within that account. That file contains a description of all optional software included on the SUBSYS tape. You should ensure it corresponds to your expectations and/or requirements. Otherwise, you may be facing a long and tedious software back-dating exercise. Do not take anything for granted

here. This is one area where HP's procedures need to be tightened up. I have seen an astonishingly high number of instances of missing or incorrect software components in the SUBSYS tape, resulting in, to put it mildly, some serious inconvenience. This should not be surprising since, to get it absolutely right, it requires close synchronization of your specific software contract with your local HP administration office and the HP Software Replication Center in California.

12. Speaking of software updates, here are some techniques for ensuring LDEV1 has the requisite amount of free space, which grows larger and larger as we speak.

Assuming you discover, usually immediately prior to your planned update, that you do not have enough contiguous disk space on ldev 1, what can you do about it?

First (alas, for Spectrum users only), there is a very nice utility from HP called CONTIGXL, which is being shipped with every new version of MPE XL (aka MPE/iX) beginning with 3.1. If you do not have it, ask the response center or some other friendly source.

This program will requires two parameters, the logical device number and the number of contiguous disk sectors you need (e.g., : CONTIGXL.PUB.SYS "-d1 -c60000" looks for 60000 contiguous sectors on ldev 1). It outputs (to $STDLIST) the files you need to purge or move to free up the space. I like to send this output to a disk file (CONTIGXL "-d1 -c60000" > myfile), manipulate myfile with my favorite editor, and then input the result to the MPEX command `ALTFILE !myfile;DEV=NOT1`. Of course, in order for this to work, you need two things: (1) MPEX, from VESOFT, and (2) the class name NOT1 configured on all system volume disk drives other than ldev 1. The second part is free and can (and should) be done with both SYSGEN and VOLUTIL. If you do not have MPEX, you can do the following to get the desired result:

```
:FILE T;DEV=TAPE
:STORE !myfile;*t
:RESTORE *t;@.@.@;DEV=NOT1;OLDDATE;SHOW=OFFLINE;CREATE
```

A note of caution here: When using VOLUTIL, once you associate a logical device number with a class, it is impossible to REMOVE it from that class without an INSTALL.

13. Still on UPDATES, a common problem I have seen recently is the refusal of NETCONTROL and/or NSCONTROL to finish successfully following an UPDATE, due to mismatching version numbers of all the software components of NS/3000.

If you did not forget to STREAM JCONFJOB.NET.SYS, your problem can be overcome by including the ;OVERRIDE option of the NETCONTROL and/or NSCONTROL commands, until you can get the problem fixed permanently. The real solution to this problem is to purge the NET.SYS group PRIOR to your the UPDATE. This way, all the NS components to which you are entitled SHOULD BE on your SUBSYS tape and be synchronized with each other (see 10). Before doing so, save your network directory file (NSDIR.NET.SYS) to some other

group.

14. We all know how difficult it is to modify and enable UDCs. (Actually, it's not really that difficult, but then the following is a good idea anyway.)

Set up a system-wide OPTION LOGON UDC (or include it in an existing one) to have the following command:

```
SETVAR HPPATH,"!!HPGROUP,PUB,PUB.SYS,CMD,CMD.SYSTEM"
XEQ LOGON.CMD.SYSTEM
```

This will add the CMD.YOURACCT and CMD.SYSTEM (or wherever you keep your globally accessible stuff) to your automatically searched path for command files and program files.

Contained in LOGON.CMD.SYSTEM are all those things that you want to happen for all users automatically each time they log on. Not only can you easily modify this file, you now have a handy repository apart from PUB.SYS -- which should really contain only HP-supported 'stuff' -- for all command files that you want to make available system-wide.

15. Here are some other useful command files:

SOJ

```
PARM JOBNO
SHOWOUT SP;JOB=#J!JOBNO
```

This shows all SPOOL files associated with a specified JOB Number (do not need #J).

SPRINT

```
PARM SPLFNO,STRTLINE=1
PRINT O!SPLFNO.OUT.HPSPOOL;START=!STRTLINE
```

The above command files work very nicely together.

For Network-related tasks, create the following command files:

NETUP

```
OPTION LIST
NETCONTROL START;NET=LOOP;OVERRIDE
NETCONTROL START;NET=LAN;OVERRIDE
NSCONTROL START;OVERRIDE
```

NETDOWN

```
OPTION LIST
NSCONTROL STOP
NETCONTROL STOP
```

NETSTAT

```
OPTION LIST
NETCONTROL STATUS
NSCONTROL STATUS
```

To purge large files quickly, use the following command file:

PURGEF

```
PARM FILENAME
FILE FILENAME=!FILENAME;ACC=OUTKEEP
PURGE *FILENAME
RESET FILENAME
```

Of course, these and many others of your liking can be placed in CMD.SYSTEM. You need not worry about security-related issues. Users still

require the appropriate capability to complete these commands.

16. Have you ever used NMMGR to reconfigure your DTC ports, revalidated, and repeated that scenario at least a couple of times (ensuring your NMCONFIG backup file does NOT reflect your current NMCONFIG), and then discovered that you really didn't need to make those changes after all, or the changes you made were all wrong?

Relax. You don't really need to perform an UPDATE CONFIG from your latest customized SLT, (You do have one of course, don't you?)

Simply log on to MANAGER.SYS, :PURGE NMCONFIG (okay, :RENAME it if you don't trust me), and :RENAME NMCONFIX,NMCONFIG. NMCONFIX represents the NMCONFIG file as it was when you last booted your system.

If you do this, you should also go into SYSGEN and do an RDCC (via the SY path).

17. This is in the category "Did You Know" or, better still, "I Wish I Had Known This Before I Upgraded." (Spectrum only.)

Prior to MPE/iX 4.0, it was possible (desirable??) to service a higher number of users than your limited user-license permitted. This was due to a technical loophole that allowed HP 3000 users to circumvent the so-called user-license limit. This was accomplished by having any "online" program be initiated via a batch job stream which simply used a file equation to specify the actual terminal device. Since no online :HELLO command was involved, it did not count as a "user" as far as the limited user license was concerned. You could purchase a 917LX instead of an identically powered, though more-concurrent-user-capable, 947 for substantially less money.

Well, guess what. Surprise!! Starting with MPE/iX 4.0, those so-called batch jobs that open terminals are considered to be actual "users" with respect to the user-license limit. In other words, the loophole has been closed.

To ascertain the maximum concurrent number of users your CPU is licensed for, simply type the following:

```
:SHOWVAR HPUSERLIMIT
```

18. Here is an easier way to manage password security for your HP 3000. Instead of assigning a different password for all your accounts, why not simply assign the same one at the account level for all those accounts that the user population normally doesn't log on to? This way, you don't have to worry about forgetting a required password. Also, you can easily change them all at the same time and on a more regular basis. This will make your system more secure than it would otherwise be. Furthermore, if you suspect that somebody has discovered a password either by accident or design, they can all be changed very quickly and simply.

This is analogous to having a building supervisor carry around a single master key that fits all doors that should not normally be entered, instead of a different key for each door, which would be more difficult to manage, and whose loss or absence would be more difficult to notice or detect.

For those of you using transaction logging for your TurboIMAGE databases,

here are some useful techniques:

19. You probably already know that you need to disable logging prior to performing any database transformation with Adager, DBGeneral, Flexibase, DBMGR, DBCHANGE or whatever tool you happen to be using.

If you then forget to re-enable logging (via DBUTIL), it's difficult to know about it. In fact, you may not even find out until such time as you really need it, only to discover it was not enabled (since that time many weeks ago when you performed that last capacity change).

How, then, can you ensure that your databases are enabled for logging?

You should include, in your partial and full backup job streams, processing steps which automatically "enable" logging for the appropriate databases.

For this you really need either MPEX (with its CHLOGON command) or XLOGON5 (Classic) or WLKABOUT (Spectrum), the latter two from the Interex Contributed Software Library. These utilities allow your backup job streams to "pseudo" logon as the database creator for each of your log-enabled databases, and run DBUTIL's ENABLE command to guarantee your databases being transaction logged.

20. One of the benefits of transaction logging is that you can reduce your daily backup tape reels by excluding logged databases from your partial backups -- as long as you ensure you back up the log files.

In order to back up the log files, you need to use ALLOWME (see above) to include the LOG logid,STOP command in your backup job streams. At the conclusion of backup, you include the appropriate LOG logid,START (or RESTART) command.

21. How do you exclude a number of databases from your partial backup? The answer is "very carefully," since you do not want to risk excluding more than you really want to, either now, or in future.

One way is to enforce a standard group naming convention which has all "logged" databases residing in a group named, for example, "DBL" (it should contain ONLY those databases). Then it's simply a matter of using the file set "@.@.@-@.DBL.@" in your partial backup.

If that is not possible, consider the following indirect store file set:

```
@.@.A@
@.@.B@
@.@.C@-OPDB@.DATA.CORP
@.@.D@
.....
@.@.Z@
```

This method ensures you do not accidentally exclude important files from your partials, because you are forced to define EXPLICITLY each file set exclusion.

As you can see, the simple application of fundamental knowledge of the HP 3000, together with a generous sprinkling of common sense, can contribute to the effective management of the HP 3000.

******

INTRODUCTION TO OBJECT-ORIENTED TECHNOLOGIES

Orland J. Larson
Hewlett-Packard Company
19091Pruneridge Avenue
Cupertino, California 95014
(408)447-1197

ABSTRACT

"Object orientation" (00) is a new buzzword that promises to become the next "eureka" in information systems development. MIS management and their software developers are under increasing pressure to improve productivity, increase software quality and reduce implementation time. Traditional software development methods are not always able to meet these increasingly heavy demands. Object orientation is getting attention as a viable alternative.

This paper reports on the growing body of knowledge about object-oriented technologies. It begins by reviewing some of the critical challenges facing today's enterprises, followed by the definitions, basic mechanisms and key concepts associated with object-oriented systems. Next, it explores various types of applications that benefit from this technology. The potential benefits and the potential concerns will be addressed, followed by the impact object-oriented technologies may have on data administration and systems development in the 90's.

INTRODUCTION

Each decade one or two key advances emerge and change the practice of software development. Object-oriented systems and methods are rapidly entering the mainstream of software engineering and systems development. Leading consultants are heralding object-oriented approaches as one of the most important trends to affect businesses in the 90's.

Software is currently lagging behind hardware capabilities and the lag is increasing. There is general agreement that conventional software tools and techniques are rapidly becoming inadequate as software systems grow larger and increasingly more complex. Also, a consensus is building that the new paradigm

of object orientation may help control complexity and harness the expanding system environment into more useful and exciting applications.

Applications will need to satisfy more sophisticated requirements, use more complex data structures and architectures, and be delivered to an increasingly broad base of users. Software developers will have to increase their capacity to build, extend, and maintain complex, large-scale systems including their existing legacy systems. This requires that software be more flexible and easier to use.

Many of today's software development processes are out-of-date, with programmers still functioning like craftsmen. They build unique, noninterchangeable components and assemble them by hand, and then they struggle over time to understand the code generated by their predecessors and to extend and refine that software. As powerful computers pervade the lives of more and more people, the inability to deliver and maintain equally powerful software is an increasingly visible problem.

WHAT IS OBJECT ORIENTATION?

There is no single precise rule for describing or identifying object orientation. Rather, a collection of concepts together describes this new paradigm for software construction. In this new paradigm objects and classes are the building blocks, while methods, messages, and inheritance produce the primary mechanisms. Historically, creating a software program involved creating processes that act on a separate set of data. Object orientation changes the focus of the programming process from procedures to objects. Objects are self-contained modules that include both the data and the procedures that act on that data. The procedures contained within the object take on a new name, methods. Objects are activated by messages. Objects that have a common use are grouped together in a class, and new classes can be created that inherit the procedures and data from classes already built. This inheritance enables the programmer to reuse existing classes and to program only the differences. This provides for a new level of abstraction, with prebuilt libraries of classes and even prebuilt application specific class libraries or frameworks. Object orientation is important for the software development challenges of the 90's. This paradigm will improve the software development process and will cause new and better applications to evolve. It's promises will be delivered incrementally and across a broad range of technologies and will permeate the next generation of software architectures.

BASIC MECHANISMS

OBJECTS

Webster's New Collegiate Dictionary defines an object as "Something that is capable of being seen, touched or otherwise sensed." Grady Booch, in his book, "Object-Oriented Design with Applications", defines an object as "Something you can do things to. An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class". David Taylor, in his book "Object-Oriented Technology: A Managers Guide" defines an object as "A software packet containing a collection of related data and methods for operating on that data".

Within objects reside the data of conventional computing languages, such as numbers, arrays, strings and records, as well as any functions, instructions, or subroutines that operate on them.

MESSAGES

Objects have the ability to act. Action occurs when an object receives a message, that is, a request, asking the object to behave in some way. When object-oriented programs execute, objects are receiving, interpreting, and responding to messages from other objects.

METHODS

Procedures called methods reside in the object and determine how the object acts when it receives a message. Methods may also send messages to other objects requesting action or information.

CLASS

Many different objects may act in very similar ways. A class is a description of a set of nearly identical objects. It is a category or collection of objects that share a common structure and a common behavior but contain different data.

INSTANCE

An instance is a term used to refer to an object that is a member of a class. Instance and object are used interchangeably.

INHERITANCE

Inheritance is the mechanism for automatically sharing methods and data among classes, subclasses, and objects. A powerful mechanism whereby classes can make use of the methods and variables defined in all classes above them on their branch of the hierarchy. Inheritance allows programmers to program only what is different from previously defined classes.

KEY CONCEPTS

ENCAPSULATION

Encapsulation is the process of hiding all of the details of an object such as its data (instance variables) and procedures (methods). This is also referred to as "information hiding".

ABSTRACTION

Abstraction is the process of creating a "superclass" by extracting common qualities or general characteristics from more specific classes or objects. Each level of abstraction makes the job of programming easier because it makes more reusable code available.

PERSISTENCE

Persistence refers to the permanence of an object, that is, the amount of time for which it is allocated space and remains accessible in the computer's memory. The object may continue to exist even after its creator ceases to exist. Objects stored permanently are termed persistent.

POLYMORPHISM

Objects act in response to the messages they receive. The same message can result in completely different actions when received by different objects. This phenomenon is referred to as polymorphism.

OBJECT-ORIENTED APPLICATIONS

Object-oriented applications will inspire users to think differently about the nature of
computing. Programs in an object-oriented environment will be transparent. Object-oriented frameworks will facilitate simulating and constructing user-specific solutions. Objects will be shared in networking environments to

distribute information within a work group or to parcel out tasks for distributed processing.

Object orientation is favored for applications that are characterized by complex processes and complex data manipulation. Applications in the following categories are classic candidates for enhancement through object orientation:

- Computer Aided Software Engineering (CASE)
- Computer Aided Instruction (CAI)
- Computer Integrated Manufacturing (CIM)
- Computer Aided Publishing (CAP)
- CAD/CAM/CAE Systems
- Document Management Systems
- Executive Information Systems
- Geographic Information Systems
- Graphics, Handling ICONS
- Health Care
- Image Storage Management
- Knowledge Based Systems
- Multimedia
- Manufacturing Production Control
- Manufacturing Requirements Planning
- Military Command and Control Decision Support
- Network Management
- Real Estate Systems
- Configuration and Version Management
- Telecommunications Routing Systems
- Visual Programming

Object-oriented applications will most likely gain in both presence and popularity.

POTENTIAL BENEFITS

Before managers can make informed decisions about adopting a new technology, the advantages of this technology must be translated into measurable benefits. Object-oriented programming improves not only the software development process but also the flexibility and utility of the resulting software. The design process becomes more intuitive as elements of the software correspond to elements in the application' s real world domain. The programming process itself encourages teamwork, code reuse, and code polishing.

Reusability is the key to increasing productivity in the face of increasing complexity. The key breakthrough in object technology is the ability to build large programs from lots of small, prefabricated ones. In addition to the increased productivity that results from reusability, using object-oriented technology can result in greater reliability because it reduces the risk of human error. Program structures remain intact, and change propagates naturally through the hierarchy of classes.

Flexibility is also a trademark of object orientation. Programmers are freed from the constraints of preestablished data types, allowing extensions of application functionality and bridging of heterogeneous applications.

Adaptability of object-oriented programs may well turn out to be the most crucial advantage of object-oriented technology. No matter how perfectly crafted, a program is useless if it doesn't meet current needs and the needs of users are changing at an ever increasing rate.

Faster development of applications is another benefit and is a result of all the programming effort that is reused from existing objects and all the design work that went into an existing model of a process.

Increased scalability is another significant benefit of object orientation. Given its improved modularization, it is especially well suited to developing large-scale systems.

Large systems are easier to build and maintain when you build them out of subsystems that can be developed and tested independently.

POTENTIAL CONCERNS

While object-oriented technology promises many benefits, there are some valid concerns about its ability to deliver those benefits. Most of these concerns have to do with temporary limitations and should disappear as the technology and its market mature.

The maturity of the technology itself is a concern to many potential developers. It is not yet a completely stable technology and many companies are not comfortable being "pioneers".

Standards are still evolving and the lack of accepted standards raises concerns about the difficulty of moving programs from one development environment to another and mixing and matching objects and classes from different vendors. Standards are on the way. The Object Management Group (OMG) was formed by a consortium of the major vendors of several object-oriented products. The purpose of this group is to promote the adoption of standards and the interchangeability of objects. In addition, the American National Standards Institute (ANSI) has an ODBMS committee, but no standards have been officially approved.

There is also a shortage of tools for application development. These tools include programs to assist in the design of objects and the management of libraries of reusable objects.

Performance of object-oriented applications is a concern and the speed of object-oriented systems will improve as the technology matures.

The object-oriented approach has a tremendous amount of potential and companies should explore this new technology, and check out the benefits for themselves.

EVOLUTION OR REVOLUTION?

Many organizations have invested heavily in existing non-relational and relational database management technology. The majority of these companies do not want to replace their existing databases and applications. The need to integrate these "legacy" databases with each other and with new systems is an important factor in the future evolution of data management.

There are clearly risks associated with getting into this technology too soon. But there are risks associated with waiting as well. The companies that begin the transition now will enjoy an important competitive advantage while the others strive to catch up.

The most prudent strategy is to avoid the extremes of ignoring the strategy or committing vital systems to it. Instead, companies can make a modest investment in a pilot program to gain first-hand experience with object-oriented development. This approach allows a company to reach its own conclusions about the value of the technology, and places the company well down the experience curve if it converts in the future.

THE FUTURE

Object-oriented technology will provide the clarity and flexibility essential to the successful development of complex systems. Today's applications do not offer the consistency and flexibility needed to make the computing environment more productive for users. Object orientation will provide environments in which users can communicate among applications and navigate easily over distributed, heterogeneous architectures.

In the near future object orientation will deliver the most benefits to three categories of programmers: power users, general business programmers and system developers. The most dramatic near-term benefits will be for system developers who both require and embrace this development approach and evolving tools to implement the increasingly complex and potentially innovative software of the 1990's. Over time, object-oriented technology will begin to have increasing impact on general business programmers and power users. Carefully designed object libraries will become available to support less sophisticated programmers who want to assemble applications quickly from prefabricated objects.

The vision of the future extends beyond the arrival of object-oriented system components development tools and standards. In the future, users will have the power and flexibility to design their own applications just by snapping together the necessary objects. With objects, building applications will be a process of tailoring and linking reusable modules. Object-oriented software architectures will mature in the 1990's. The transition to these new architectures is underway, marked by the arrival of object-oriented languages, databases, interfaces, operating systems and development environments. New types of data, distributed processing, multimedia applications, and end user computing are driving forces in the implementation of the object-oriented software environment.

There has been a great deal of progress implementing object-oriented systems. Today's graphical user interfaces have acquainted users with object manipulation. Among object-oriented languages, C++ has become the de facto standard. Object-oriented extensions are also being implemented in most popular commercial languages. Object-oriented development environments such as Hewlett-Packard's Softbench provide examples of object-oriented programming and applications. Operating systems are also being extended to support interoperability among object-based applications. Over the next decade, the difference between the old and the new will become increasingly obvious to both programmers and users. When the object-oriented future is fully delivered, this

natural, intuitive paradigm will be strongly embraced and will provide benefits to programmers and users alike.

OBJECT TECHNOLOGY FROM HEWLETT-PACKARD

Hewlett-Packard recently announced HP OpenODB, the most advanced, commercial object-oriented database management system for large, multi-user environments. It is designed to enable new, complex business applications to be developed and maintained at a fraction of current costs.

HP OpenODB is based upon the Iris ODBMS prototype developed by HP Labs, starting in 1984. Using Iris, HP has worked extensively with customers and universities in evaluating ODBMS requirements. HP OpenODB is targeted at complex, commercial applications such as Geographic Information Systems (GIS), telecommunications systems and heterogenous information integration such as Executive Information Systems and Computer Integrated Manufacturing, whose needs may not be met by current database products such as TurboIMAGE and ALLBASE/SQL.

HP OpenODB uses a relational database as its storage manager and presents an object-oriented model to developers. It is a hybrid approach that allows integration of existing legacy systems including applications written in C, COBOL, FORTRAN, PASCAL, ADA, etc. This architecture provides a robust storage environment with the DBMS capabilities commercial users have come to expect. HP is currently using ALLBASE/SQL as the storage manager for HP OpenODB for performance and stability of data; however, the architecture allows HP OpenODB to be ported to other relational DBMS's for portability to non HP hardware. This combination is unique in the industry.

A Key benefit of HP OpenODB is that users don't have to abandon their current software or data to work with it. It includes an object-oriented structured query language (OSQL) and external functions that will retrieve information regardless of its format or whether it is stored inside or outside of HP OpenODB.

Object orientation is another phase in the evolution of computing and is an important step towards the vision of "Information At Your Fingertips". Developers must take advantage of the many benefits of this technology while at the same time deal with the hurdles that this technology poses. Object-oriented development environments must play a large role in reducing the learning curve and make object-oriented programming a highly productive process. Object-oriented products are here today and the commercialization of

object-oriented technology is increasing rapidly. Object based architectures lend themselves to the creation of a much richer information environment. Digitized voice, music, video clips and animation will begin to populate our information systems. Object database systems are currently viable for commercial projects and will be widely adopted by the mid to late 1990's.

REFERENCES

Booch, Grady. *Object-Oriented Design With Applications.* Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc, 1990.

DeCastilhos,Margie.*Getting Started With Object-Oriented Databases.* INTERACT Magazine (Hewlett-Packard User Group Publication).pages 102-113, March 1992.

Hewlett-Packard.1990. *CASE and Objects-A Primer(An Introduction to Object-Oriented Concepts for Software Engineering),* Palo Alto , CA:HP Literature # 5952-2624.

Taylor, David A., *Object-Oriented Technology: A Manager's Guide.*Reading,MA. Addison-Wesley,1991. (A very good overview of object-oriented technologies for managers)

Winblad, Ann L., et al. *Object-Oriented Software.*Reading, MA. Addison-Wesley 1990.

THINK
ABOUT
USING
OBJECTS!

INCORPORATING DYNAMIC ELECTRONIC FORMS INTO HP 3000 APPLICATIONS

Ross G. Hopmans
Brant Technologies Inc.
51 Tannery Street
Mississauga, Ontario
Canada L5M 1V3
(416) 858-0153

The purpose of this paper is to demonstrate how to take full advantage of the advanced capabilities of the HP LaserJet in your production applications. I will accomplish this by demonstrating how my company has integrated the FANTASIA/3000 laser printing software into our internal sales and marketing system on the HP 3000.

FANTASIA/3000 is from Proactive Systems and most of the functionality I will be addressing in this paper is also available in their FANTASIA/UX software for the HP 9000.

PREMISE

The use of electronic forms only scratches the surface of how we can enhance our production reports with the LaserJet. Electronic forms are usually a static replacement for pre-printed forms. The use of these static electronic forms is a good first step. It can represent significant cost savings as well as added convenience with no changes to your source code.

The reference in the title of this paper to *dynamic* electronic forms means that you can incorporate typesetting technology into your production reports so that the forms you produce are dynamic – a form can adapt itself to the nature and amount of data that your application provides. Most importantly, this all takes place on the HP 3000 as part of your production runs. Typesetting technology allows us to go beyond electronic forms to produce much more effective laser output.

This paper focuses on the ability to *typeset* production data on the HP 3000

so that all of your reports have a professional, customized, typeset look. Although this technique involves changes to the source code of your application programs, the quality and effectiveness of the results easily justify the effort. In addition, I maintain that for new reports it is actually less work for programmer to take advantage of typesetting software than it is to produce a report in the more traditional manner.

BACKGROUND

Brant Technologies is a small company with a strong commitment to professionalism in all aspects of our business. In addition, we strive to be as highly automated and efficient as possible with minimized administrative overhead. The use of FANTASIA/3000 has helped us accomplish both goals.

The lead tracking and mail merge functions of our SPEEDWARE-based sales and marketing system had been evolving for several years as our needs evolved. The system incorporates an OMNIDEX interface to allow sales and marketing personnel to locate individuals or groups of people in our data base using many search criteria. Essentially the system contains contact information, prospect data for sales forecasting and notes on the history of client contact. We have since expanded this application into a sales system as well to generate invoices, commission statements, royalty reports, sales history and more. All reports are now produced with FANTASIA/3000.

Giving a typeset, customized appearance to all of our printed output reflects well on the company. It makes us look professional while avoiding the costs associated with professionally typeset documents. But in addition, as soon as we enter an order to the system, we generate the form to order the product from our supplier, shipping documentation, an envelope (or label) and, most importantly, an invoice. There are no special forms to mount and specific output is directed to the appropriate LaserJet. All of this is driven by a salesperson entering a customer order. There is nothing to impede the process.

We will now examine the types of reports this system produces and what is involved in their production. I have also included some other examples I have developed to give the reader a more complete picture of the capabilities that are available.

REPORTS

Let's begin by taking a look at some of the output that our SPEEDWARE application produces.

Figure 1 is a typical invoice. This is the administration copy. You can see that we have had our logo scanned and we incorporate it into the invoice. In fact, the customer copy is printed on letterhead for the color and higher-quality paper. We print all subsequent copies with the scanned logo. Each copy has its destination identified at the bottom. The customer copy does not show the GL Code field so the Description field is wider.

Note that I am using proportional fonts throughout. The invoice has a much nicer look than the Courier or "typewriter" font of impact printers. In addition, the gap between the label and the data for Invoice Number, Purchase Order and Invoice Date is proportionately filled with dots for a customized look. Perhaps more importantly, the description field is typeset by FANTASIA/3000 and not by my application program. The program simply hands three 60-column fields of description data to FANTASIA/3000 and it is FANTASIA/3000 that decides how much will fit on a line and where to break the line. I no longer have to program that logic into my application. The same is true for Terms. The terms are "Net thirty days" and the Currency is "Canadian". My application gives FANTASIA/3000 four lines of data: the terms, the literal "in", the currency and the literal "funds.". FANTASIA/3000 puts these into a sentence and I am saved from having to format these in my application. So part of this form is static (ie. the grids and labels) and part is dynamic (ie. the data provided by the application).

If I had been using pre-printed forms, I could have replaced them very easily with "static" electronic forms, making no source changes to my application. Figure 2 shows what that would look like. Compare Figure 2 to Figure 1 and see which look you prefer. No matter which you choose, the use of electronic forms amounts to approximately 1/3 the cost of pre-printed forms, not including the extra costs associated with the initial design and setup of the pre-printed form, the waste, mounting, bursting, decollating, storing and bulk purchases of pre-printed forms.

There are also costs and time involved in form changes. Since implementing this system, we have had the GST (Goods and Services Tax) imposed in Canada and later this year we will go through an area code change to our telephone number. Both of these changes have significant impact on forms. By using electronic forms I avoid the expense of ordering new forms and scrapping the old.

Figures 3 and 4 are examples of sales commission reports. The first, Figure 3, was one of the first forms I developed using FANTASIA/3000. It looks very much like a replacement for a pre-printed form because the grids are static. Although it is a very attractive form, I have since changed the application to print Figure 4. This is truly a dynamic form where the data itself determines how large the grid area is. It easily spans multiple pages and the total box only appears at the end of the form. The most important

data item on the page is the total commission paid and that is highlighted in a drop box with a larger font. We will take a look at the source code changes that I made to implement this report later in the paper.

Incidentally, this may not be a good example of a report which is cost justified to replace with enhanced laser technology. Typically, reports such as sales commission are internal and do not use pre-printed forms. But I maintain that all of your company's reports are important. If your report has a professional, customized look to it, it reflects well on the people who produced the report and it makes a difference to the recipient. Our sales people love their commission reports. It is important for them to feel that we appreciate their results and, quite frankly, this is a very easy report to produce.

Figure 5 is a fax message. I was unhappy with the quality of correspondence being faxed out of our office. We had a standard fax cover sheet to hand-write enclosure information, but these were usually illegible. Since most faxes are sent to people already in our data base (which has to be accessed anyway to check for the fax number) I added the capability to produce formatted faxes quite easily. Once the user selects the individual to whom he or she wants to address the fax, they select "FAX" from the report menu. This calls our favourite editor to allow the user to type the fax message. Once complete, the user simply exits from the editor and the system calls FANTASIA/3000 to process the FAX form and pull in their message text. It is simple, easy, fast and it gives a consistent, professional image to all material being faxed from our office.

Figure 6 is an example of archive output which I use solely to save paper. This shows a tape validate listing such as I keep for my backups. FANTASIA/3000 picks up all spool files named LIST (as well as SYSLIST, $STDLIST and others) and applies an "environment" file to print them four-up with boxes around each logical page. Note that I am only showing two pages up in portrait mode to fit in the proceedings. However the actual output appears in landscape mode with each of the boxes show taking up one quarter of the page. If I had a duplex printer, I could print on both sides of the paper which would give me eight logical pages on one physical piece of paper. I save paper, I save trees, and I take up less space in my binder of file listings. All of this adds up to saving money.

Figures 7 and 8 show before and after examples of a price list driven from a data base on the HP 3000. We do not publish price lists ourselves; this is an example produced for someone else. It is a good demonstration of how to incorporate typesetting into your production reports. Figure 7 shows the type of report that you can produce on the LaserJet using the default Courier font. Many companies produce catalogs and other reports that are sent out to be professionally typeset at considerable expense. Figure 8 demonstrates

the enhanced output that could be created on the HP 3000 in a production environment. The nature of the data lends itself to being printed two-up in landscape mode. The header and footer data extend across the entire page. Perhaps more subtly, the footer area contains a number of terms, each of which is an entry in a TERMS data set. FANTASIA/3000 dynamically formats the footer area to leave enough room for all of the terms. So we get this professional look all driven from our production data and the formatter does much of the work for us.

My final example, Figure 9, shows a mail merge generating word processed-quality letters. These use "boiler plate" or standard text with variables such as name, address, salutation, etc. extracted from our corporate data bases. So the resulting letters all have the quality of a word processor but we are able to keep everything on the HP 3000 so that we can have large, production runs and treat them like any other report. Moreover, we can incorporate scanned signatures to save the time and expense of hand signing each letter.

This is not meant to be an exhaustive display of our enhanced reports; nor does it begin to cover all you can do with your LaserJet. However, it does represent a cross-section of the types of reports that we produce and some of the important output capabilities that many HP 3000 shops require.

INTEGRATION

Let's now take a look at how I have created two of these reports. Figure 10 shows a progression of my commission report from unformatted to simple formatting to the finished product. In order to accomplish this, I alter the source code to my application to embed FANTASIA/3000 formatting commands in with the data itself.

FANTASIA/3000 takes a post-processing approach. Your application creates output in the spooler or as flat files. FANTASIA/3000 then processes that output (calling "include" files, graphics, static forms, etc. as required) to create a spooler file of enhanced LaserJet output.

Figure 11 shows the file that produces the enhanced commission report in Figure 4. Note that it contains both data and FANTASIA/3000 commands – either a command line starting with a "\" in column 1 or delimited by "^" within data lines. The FANTASIA/3000 commands are created as literals in my application program. So the enhanced output can be created through Cobol, 4GLs such as SPEEDWARE and POWERHOUSE and even report writers which can include literals.

The body of the report is contained within the TABLE command. TABLE

formats data into columns in a very flexible way. I have specified that the table contains 5 columns with column 2 a particular width. I specify the justification for the individual columns, shading if required and individual widths if I want to override the defaults. The actual data items can be separated by a specified delimiter, by at least two blanks, in specified column positions or one data item per line. The FANTASIA/3000 formatter will ensure that rows line up and it looks after breaking individual columns into multiple rows.

As a programmer, I have found that there are three important differences in creating output using FANTASIA/3000 over traditional methods (ignoring the quality and effectiveness of the output).

The first is that you have become married to your LaserJets. In most organizations that may not be an important point. But it is worth mentioning that if your LaserJet breaks, you can no longer redirect your FANTASIA/3000 output to your impact printers. It will not work.

On the plus side, the second difference is that the programmer now operates at a higher level. You no longer tell your application exactly how to print your data. You operate more as a typesetter and you instruct your application in how to process the data. If you want the data printed in a twelve point helvetica font, justified on both right and left margins and wrapped to fill fill all lines with 14 point line spacing in an area 3 inches wide measured 1 inch from the left margin you would issue the following commands:

```
\fontset h scale helv 12
\left 1;width 3;just both;font h;wrap;linespace 14
```

Programming is more productive because the FANTASIA/3000 formatter can do most of the work. FANTASIA/3000 looks after the logic involved in spacing and how much data can fit on each line, and so on. All we have to do is define the environment to FANTASIA/3000 and then give it the data.

My third point is that I now prototype my reports outside of my application development environment, be it SPEEDWARE, COBOL or whatever. I work with MPE files containing dummy data until I get the report to look exactly the way I want. Then I simply program my application to create output that looks like my prototype file. This is a real time saving because I do not have to go through the recompile, select, sort, etc. for each iteration of report refinement.

Figure 12 shows the MPE file containing the standard text and formatting directives to produce the mail merge letter in Figure 9. You can see how I have positioned and called the company logo at the end of the file. Although

I have not included it here, I could have brought in a signature as well to produce ready-to-mail output.

This file contains formatting directives that define the fonts, margins, spacing, etc. In addition, it says that every time we see the macro "a" (as "^ma") in the letter, replace it with the next data item from the file "datafile" and repeat until we exhaust all the data in the file. This assumes that we have created an extraction file of of selected entries in the file "datafile". We could have done that with QUIZ, a report writer or any other means. We then run FANTASIA/3000 against this letter file to access that data and format the letter dynamically. This gives us a word processed look in our production environment on the HP 3000. And of course we can process one letter or thousands of letters once we have selected the appropriate data.

CONCLUSION

I have given you a taste of some of the enhanced LaserJet output we produce in our environment and have shown that electronic forms are much less expensive and much more convenient to use than pre-printed forms. Enhanced LaserJet reporting through the use of typesetting produces more effective reports that give our company a high-quality, professional image.

To make the best use of this technology, we made changes to the source code of our application programs. This is not necessary to simply replace pre-printed forms with static electronic forms. However, we made the source code changes to achieve a customized, typeset look in all of our producation reports. Although this represents an additional time investment, FANTASIA/3000 is not difficult to learn and we have been able to take advantage of the productivity aspects of the product.

The net result has been a dramatic improvement in the quality of our output. We produce better, more effective reports that reflect well on the whole organization.

Ross G. Hopmans
1993.06.01

INVOICE

invoice to:
Accounts Payable
Widget Manufacturing Company
51 Any Street
Mississauga, Ontario
L4W 4L5

notes:
Shipped May 9, 1991 on DAT Media

invoice number BTC400

purchase order 1234

invoice date 08-MAY-91

qty	*description*	*gl code*	*unit price*	*discount*	*extension*
1	Widget Software for HP 3000 Series 949 (Primary CPU)	21-1410	5,000.00	N/A	5,000.00
1	Widget Software for HP 3000 Series 922 LX (Secondary CPU)	21-1410	3,000.00	50 %	1,500.00

Terms:
Net thirty days payable in Canadian funds.

Remit to:
Brant Technologies Inc.
51 Tannery Street
Mississauga, Ontario
CANADA L5M 1V3
tel (416) 858-0153

Shipping	50.00
Sub-Total	6,550.00
G.S.T.	458.50
P.S.T.	524.00
TOTAL	$7,532.50

ADMINISTRATION COPY

Dynamic Electronic Forms - **FIGURE 1**
6001 – 8

INVOICE

invoice to:
Accounts Payable
Widget Manufacturing Company
51 Any Street
Mississauga, Ontario
L4W 4L5

notes:
Shipped May 9, 1991 on DAT

invoice number BTC400

purchase order 1234

invoice date ... 08-MAY-91

qty	*description*	*gl code*	*unit price*	*discount*	*extension*
1	Widget Software for HP 3000 Series 949 (Primary CPU)	1410	5,000	N/A	5,000
1	Widget Software for HP 3000 Series 922 LX (Secondary CPU)	1410	3,000	50 %	1,500

Terms:
Net thirty days.

Shipping	50
Sub-Total	6,550
G.S.T.	458
P.S.T.	524
TOTAL	$7,532

Remit to:
Brant Technologies Inc.
51 Tannery Street
Mississauga, Ontario
CANADA L5M 1V3
tel (416) 858-0153

ADMINISTRATION COPY

Dynamic Electronic Forms – **FIGURE 2**
6001 – 9

SALES COMMISSION

Jane Doe
June 1992

invoice	customer	product	full amount	commission
C0532	University of New Brunswick	KappaPC	1,150.00	57.50
		KappaPC Sup	1,050.00	52.50
C0547	Fraser Inc.	KappaPC	4,250.00	212.50
		KappaPC Sup	1,050.00	52.50
		KappaPC Run	550.00	27.50
C0552	Pulp and Paper Research Institute	KappaPC	4,250.00	212.50
		KappaPC Sup	995.00	49.75
C0609	Concordia University	KappaPC	1,150.00	57.50
Total Commission				$722.25

Dynamic Electronic Forms – **FIGURE 3**
6001 – 10

BRANT TECHNOLOGIES INC.

Commission Report - June 1992

Salesperson: **Jane Doe**

Invoice	Company	Product	Price	Comm
C0532	University of New Brunswick	KappaPC	1,150.00	57.50
		KappaPC Sup	1,050.00	52.50
C0547	Fraser Inc.	KappaPC	4,250.00	212.50
		KappaPC Sup	1,050.00	52.50
		KappaPC Run	550.00	27.50
C0552	Pulp and Paper Research Institute	KappaPC	4,250.00	212.50
		KappaPC Sup	995.00	49.75
C0609	Concordia University	KappaPC	1,150.00	57.50

$722.25

Dynamic Electronic Forms – **FIGURE 4**
6001 – 11

Brant Technologies Inc.

FACSIMILIE MESSAGE

fax: (416) 542-1766

to:	Barry Henderson Corfax Systems	**from:**	Ross Hopmans
fax:	(416) 868-6666	**date:**	July 29, 1993

This is the first of 1 page(s).

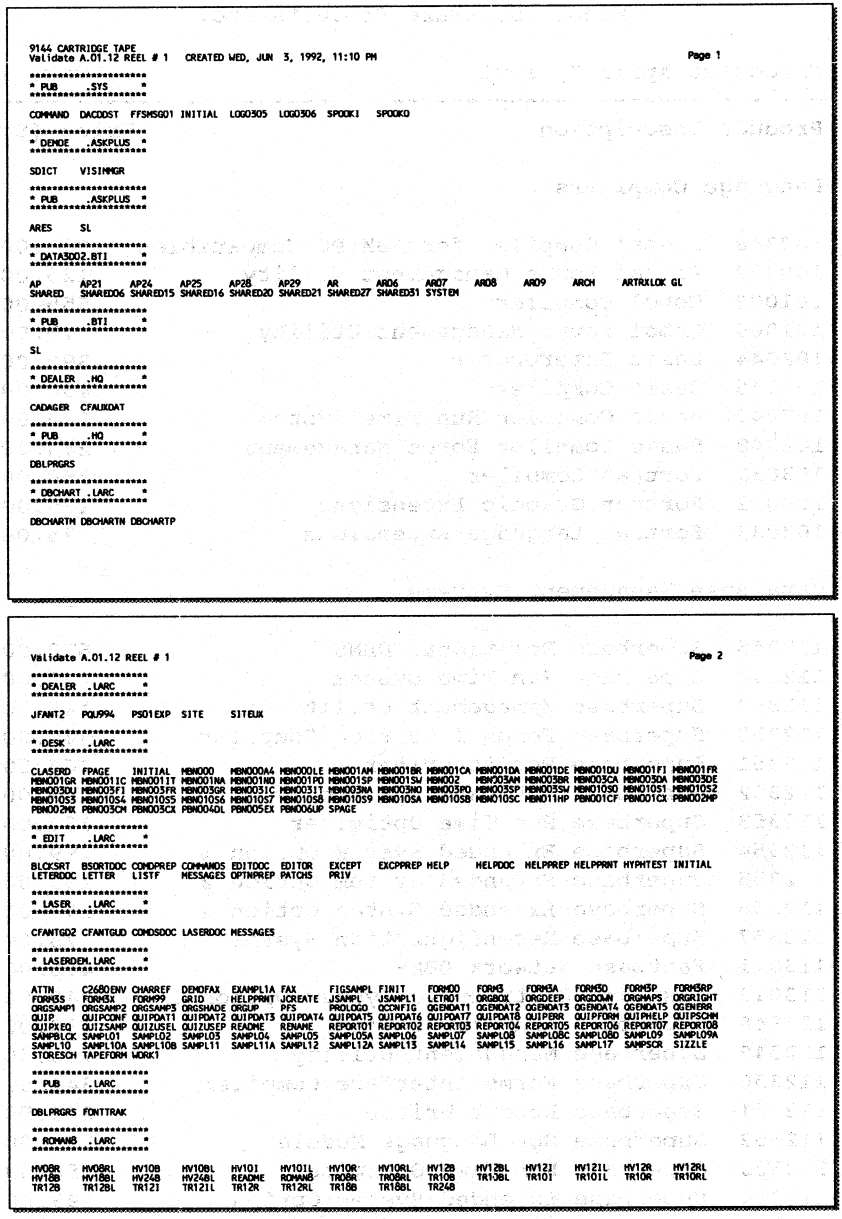
Barry,

Thanks for the opportunity to speak at SIG SPEEDWARE. I propose calling the presentation "Integrating Speedware and Fantasia - Reports That Are Dressed To Kill".

Please let me know if you have any questions. Could you also be sure that my name is on your list to notify of meetings and topics.

Thank you.

Dynamic Electronic Forms – **FIGURE 5**
6001 – 12



```
9144 CARTRIDGE TAPE                                                    Page 1
Validate A.01.12 REEL # 1   CREATED WED, JUN  3, 1992, 11:10 PM

**********************
* PUB      .SYS      *
**********************

COMMAND  DACDDST  FFSMSG01 INITIAL  LOGO305  LOGO306  SPOOKI   SPOOKO

**********************
* DEMOE    .ASKPLUS  *
**********************

SDICT    VISIMMGR

**********************
* PUB      .ASKPLUS  *
**********************

ARES     SL

**********************
* DATA3002.BTI       *
**********************

AP       AP21     AP24     AP25     AP28     AP29     AR       AR06     AR07     AR08     AR09     ARCH     ARTRXLOK GL
SHARED   SHARED06 SHARED15 SHARED16 SHARED20 SHARED21 SHARED27 SHARED31 SYSTEM

**********************
* PUB      .BTI      *
**********************

SL

**********************
* DEALER   .HQ       *
**********************

CADAGER  CFAUXDAT

**********************
* PUB      .HQ       *
**********************

DBLPRGRS

**********************
* DBCHART  .LARC     *
**********************

DBCHARTM DBCHARTN DBCHARTP
```

```
Validate A.01.12 REEL # 1                                              Page 2

**********************
* DEALER   .LARC     *
**********************

JFANT2   PQU994   PS01EXP  SITE     SITEUK

**********************
* DESK     .LARC     *
**********************

CLASERD  FPAGE    INITIAL  MBN000   MBN000A4 MBN000LE MBN001AM MBN001BR MBN001CA MBN001DA MBN001DE MBN001DU MBN001FI MBN001FR
MBN001GR MBN001IC MBN001IT MBN001NA MBN001ND MBN001PG MBN001SP MBN001SW MBN002   MBN003AM MBN003BR MBN003CA MBN003DA MBN003DE
MBN003DU MBN003FI MBN003FR MBN003GR MBN003IC MBN003IT MBN003NA MBN003NO MBN003PO MBN003SP MBN003SW MBN010SO MBN010S1 MBN010S2
MBN010S3 MBN010S4 MBN010S5 MBN010S6 MBN010S7 MBN010S8 MBN010S9 MBN010SA MBN010SB MBN010SC MBN011HP PBN001CF PBN001CX PBN002MP
PBN002MX PBN003CM PBN003CX PBN004DL PBN005EX PBN006UP SPAGE

**********************
* EDIT     .LARC     *
**********************

BLCKSRT  BSORTDOC COMDPREP COMMANDS EDITDOC  EDITOR   EXCEPT   EXCPPREP HELP     HELPDOC  HELPPREP HELPPRNT HYPHTEST INITIAL
LETERDOC LETTER   LISTF    MESSAGES OPTNPREP PATCHS   PRIV

**********************
* LASER    .LARC     *
**********************

CFANTGD2 CFANTGUD COMDSDOC LASERDOC MESSAGES

**********************
* LASERDEM.LARC      *
**********************

ATTN     C2680ENV CHARREF  DEMOFAX  EXAMPL1A FAX      FIGSAMPL FINIT    FORM00   FORM3    FORM3A   FORM5D   FORM5P   FORM5RP
FORM5S   FORM5X   FORM99   GRID     HELPPRNT JCREATE  JSAMPL   JSAMPL1  LETR01   ORGBOX   ORGDEEP  ORGDOWN  ORGMAPS  ORGRIGHT
ORGSAMP1 ORGSAMP2 ORGSAMP3 ORGSHADE ORGUP    PFS      PROLOGO  QCCNFIG  QEENDAT1 QEENDAT2 QEENDAT3 QEENDAT4 QEENDAT5 QEENERR
QUIP     QUIPCONF QUIPDAT1 QUIPDAT2 QUIPDAT3 QUIPDAT4 QUIPDAT5 QUIPDAT6 QUIPDAT7 QUIPDAT8 QUIPERR  QUIPFORM QUIPHELP QUIPSCHM
QUIPXEQ  QUIZSAMP QUIZUSEL QUIZUSEP README   RENAME   REPORT01 REPORT02 REPORT03 REPORT04 REPORT05 REPORT06 REPORT07 REPORT08
SAMPBLCK SAMPL01  SAMPL02  SAMPL05  SAMPL06  SAMPL09  SAMPL09A SAMPL10  SAMPL10A SAMPL10B SAMPL11  SAMPL11A SAMPL12  SAMPL12A
SAMPL13  SAMPL14  SAMPL15  SAMPL16  SAMPL17  SAMPSCR  SIZZLE   STORESCH TAPEFORM WORK1

**********************
* PUB      .LARC     *
**********************

DBLPRGRS FONTTRAK

**********************
* ROMAN8   .LARC     *
**********************

HV08R    HV08RL   HV10B    HV10BL   HV101    HV101L   HV10R    HV10RL   HV12B    HV12BL   HV12I    HV12IL   HV12R    HV12RL
HV18B    HV18BL   HV24B    HV24BL   README   ROMAN8   TR08R    TR08RL   TR10B    TR10BL   TR10I    TR10IL   TR10R    TR10RL
TR12B    TR12BL   TR12I    TR12IL   TR12R    TR12RL   TR18B    TR18BL   TR24B
```

Dynamic Electronic Forms – **FIGURE 6**
6001 – 13

Widget Software Products Inc.

Effective April 9, 1991 Page 1

Product	Description	Price ($)
	Language Compilers	
100348	Pascal Compiler for IBM PC Compatibles	340.00
100352	Pascal Forms Management Utility	125.00
101008	Cobol Compiler	555.00
101009	Cobol Forms Management Utility	75.50
102044	Basic Interpreter	390.00
102046	Basic Compiler	450.00
102047	Basic Compiler Run Time System	199.00
102048	Basic Compiler Forms Management	240.00
103031	Fortran Compiler	350.00
103032	Fortran Graphic Extensions	199.00
103033	Fortran Language Extensions	75.00
	Data Base Management Systems	
112345	Superbase Relational DBMS	670.00
112348	Superbase Run Time System	230.00
112349	Superbase Management Utility	145.50
112350	Superbase Forms Interface Compiler	340.00
112351	Superbase Report Writer	265.50
112352	Superbase SQL Language Module	288.00
112353	Superbase Run Time Optimiser	350.00
112354	Superbase Extended System Option 1	199.00
112355	Superbase Extended System Option 2	199.00
112356	Superbase Extended System Option 3	199.00
112357	Superbase Reconfiguration System	75.60
113011	Fastbase Network DBMS	475.00
113012	Fastbase Distributed System Option 1	240.00
112348	Superbase Run Time System	230.00
112349	Superbase Management Utility	145.50
112350	Superbase Forms Interface Compiler	340.00
112351	Superbase Report Writer	265.50
112352	Superbase SQL Language Module	288.00
112353	Superbase Run Time Optimiser	350.00
112354	Superbase Extended System Option 1	199.00
112355	Superbase Extended System Option 2	199.00
112356	Superbase Extended System Option 3	199.00

Widget Software Products Inc.

Effective April 9, 1991

Product No.	Description	Price ($)
Language Compilers		
100348	Pascal Compiler for IBM Compatibles	340.00
100352	Pascal Forms Management Utility	125.00
101008	Cobol Compiler	555.00
101009	Cobol Forms Management Utility	75.50
102044	Basic Interpreter	390.00
102046	Basic Compiler	450.00
102047	Basic Compiler Run Time System	99.00
102048	Basic Compiler Forms Management	240.00
103031	Fortran Compiler	350.00
103032	Fortran Graphic Extensions	99.00
103033	Fortran Language Extensions	75.00
Data Base Management Systems		
112345	Superbase Relational DBMS	670.00
112348	Superbase Run Time System	230.00
112349	Superbase Management Utility	145.50
112350	Superbase Forms Interface Compiler	340.00
112351	Superbase Report Writer	265.50
112352	Superbase SQL Language Module	288.00
112353	Superbase Run Time Optimiser	350.00
112354	Superbase Extended System Option 1	199.00
112355	Superbase Extended System Option 2	199.00
112356	Superbase Extended System Option 3	199.00
112357	Superbase Reconfiguration System	75.60
113011	Fastbase Network DBMS	475.00
113012	Fastbase Distributed System Option 1	240.00
113013	Fastbase Distributed System Option 2	240.00
114012	Superbase/2 Relational DBMS	670.00
114013	Superbase/2 Run Time System	230.00
114014	Superbase/2 Management Utility	145.50
114350	Superbase/2 Forms Interface Compiler	340.00
114351	Superbase/2 Report Writer	265.50
114352	Superbase/2 SQL Language Module	288.00
114353	Superbase/2 Run Time Optimiser	350.00
114354	Superbase/2 Extended System Option 1	199.00
114355	Superbase/2 Extended System Option 2	199.00
114356	Superbase/2 Extended System Option 3	199.00
114357	Superbase/2 Reconfiguration System	75.60
115011	Fastbase/2 Network DBMS	475.00
115012	Fastbase/2 Distributed System Option 1	240.00
115013	Fastbase/2 Distributed System Option 2	240.00

Prices are subject to change without notice. Prices include delivery. Contact your sales representative for a specific quotation.

Dynamic Electronic Forms – FIGURE 8
6001 – 15

May 2, 1992.

Mr. Mike Tree
Widget Manufacturing Corp.
123 Any Street
Mississauga, Ontario
L7N 1H8

Dear Mr. Tree,

Please be aware that your debt of $1,052.93 is woefully overdue at our office. We have not heard from you since April 1, 1990. Despite numerous reminders by telephone and letter, we have received no response.

Unless your money is in our hands within 10 days we will repossess your garden shed.

Have a nice day.

Yours truly,

R.E. Grinch

Dynamic Electronic Forms – **FIGURE 9**
6001 – 16

WIDGET SOFTWARE PRODUCTS
Commission Report - April 1991

Salesperson: Jane Doe

Invoice	Company	Product	Price	Comm
ABC123	First National Bank	Widgets	15,000	750.00
ABC124	Jim's Garage	Widgets	23,000	1,150.00
ABC124	Jim's Garage	Tie Rods	7,500	375.00
ABC125	Orange County	I Beams	1,200	60.00
ABC125	Orange County	Widgets	14,500	725.00
	TOTAL			$3,060.00

WIDGET SOFTWARE PRODUCTS

Commission Report - April 1991

Salesperson: Jane Doe

Invoice	Company	Product	Price	Comm
ABC123	First National Bank	Widgets	15,000	750.00
ABC124	Jim's Garage	Widgets	23,000	1,150.00
ABC124	Jim's Garage	Tie Rods	7,500	375.00
ABC125	Orange County	I Beams	1,200	60.00
ABC125	Orange County	Widgets	14,500	725.00

WIDGET SOFTWARE PRODUCTS

Commission Report - April 1991

Salesperson: **Jane Doe**

Invoice	Company	Product	Price	Comm
ABC123	First National Bank	Widgets	15,000	750.00
ABC124	Jim's Garage	Widgets Tie Rods	23,000 7,500	1,150.00 375.00
ABC125	Orange County	I Beams Widgets	1,200 14,500	60.00 725.00

$3,060.00

Dynamic Electronic Forms – **FIGURE 10**
6001 – 17

At Start of Report ...

```
\port; left 1; width 6; nowrap; just left; top 1.55; thick 3; shade 50
\linespace 9; just left
\if III skip 2
\fontset t 19, r 5, i 15, b 16
\skip 4
\fontset t scale helv 18 bold
\fontset r scale times 10
\fontset i scale helv 10 ital
\fontset b scale helv 10 bold
\head begin
^ft^^nc^BRANT TECHNOLOGIES INC.

^fb^^nc^Commission Report - June 1992

\head end
```

At Start of Salesperson ...

```
^fi^Salesperson:  ^fb^Jane Doe
\table 5 just left col 1 just center col 2 width 2.2 horiz 0 &
\table col 4 just right col 5 just right col 5 shade 2 &
\table col 1 width 0.6 col 3 width 1.0
\fillbox 0.25; font b
^nc^Invoice  ^nc^Company  ^nc^Product  ^nc^Price  ^nc^Comm^fr^
```

At Start of Invoice ...

```
\line 8
```

At Detail Time ...

```
C0532  University of New Brunswick      KappaPC       1,150.00   57.50
                                        KappaPC Sup   1,050.00   52.50
\line 8
C0547  Fraser Inc.                      KappaPC       4,250.00  212.50
                                        KappaPC Sup   1,050.00   52.50
                                        KappaPC Run     550.00   27.50
\line 8
C0552  Pulp and Paper Research Institute KappaPC       4,250.00  212.50
                                        KappaPC Sup     995.00   49.75
\line 8
C0609  Concordia University             KappaPC       1,150.00   57.50
```

At End of Salesperson ...

```
\tableend
\inleft 5.05; dropbox 0.4 0.1

\space inch 0.1; just center; font b
$722.25
```

```
\left 1.5; width 5; top 2; just left; nowrap; font 8
\fillin macro=a, file=datafile, repeat
^d1^.

^ma
^ma
^ma
^ma
^ma
\space inch 0.25
Dear ^ma,

\indent 0.5; wrap
Please be aware that your debt of ^ma is woefully
overdue at our office.  We have not heard from you
since ^ma.  Despite numerous reminders by
telephone and letter, we have received no
response.

Unless your money is in our hands within ^u10 days^s
we will repossess your ^ma.

Have a nice day.

\indent off; nowrap
Yours truly,



R.E. Grinch

\goto 0.16 0.1
\graph brant1p.pub
```

Paper 6003

The Systems Management Service Partnership

Kyle S. Adler
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043
(415) 691-5081

Introduction

Today more than ever before, Information Technology (IT) departments are faced with challenges in providing effective systems management (SM) services. A host of new pressures is increasing the systems management workload, including user expectations of increased service levels, migration to open systems, and client/server implementations. Simultaneously, a set of different factors is serving to reduce the resources at the IT manager's disposal; such facts of life in the 1990s as budget reductions and caps, fixed or shrinking headcounts, and data center consolidation can siphon off IT's human and capital means to attack the new challenges. What is resulting is a fundamental paradox: How can IT manage more with fewer resources?

Many organizations are finding that partnering with outside service providers allows their systems managers to do more with less. IT partnership with outside vendors is by no means new, and indeed, very few if any IT departments accomplish all tasks using in-house resources alone. However, the new challenges facing IT departments suggest new ways of partnering, to the mutual benefit of IT and the outside service provider.

This paper will suggest ways to make the partnership work. We will explore the trends contributing to the new challenges in order to find some of the daily operational tasks that can be shared with an outside service provider, examine the criteria to consider--including financial analysis--when deciding how to selectively partner with a service firm, and outline proven methods to ensure a win-win partnership.

Why is the Job Becoming Harder?

In the 1980s a systems manager had to perform very nearly the same tasks as today. Back then as now, problems had to be detected, isolated, and resolved; jobs had to be scheduled and executed; data backed up and recovered; security maintained; system and network configurations

managed; and so on. Furthermore, most of the types of computing equipment--mainframes, minicomputers, PCs, workstations, LANs, WANs, and peripherals--found in organizations today were also present a decade ago, albeit used in very different ways. So what has changed in IT departments over the last decade to cause such a crunch on resources? Let us examine first the trends contributing to the reduction of IT resources, followed by a look at why the workload is growing at an exponential pace. Together, these trends create the paradox of expecting to do more with less.

Factors Reducing IT Resources

The buzzwords are peppered through business headlines in every major journal: "downsizing", "rightsizing", "consolidation", "global competition". Corporate downsizing is a business reality in the 1990s. Because IT represents a large and growing portion of most organizations' budget, CFOs often look to their IT department first when deciding where to slash budgets. Under pressure that often begins as high as the CEO and Board of Directors, CFOs and CIOs are demanding greater return on the company's IT investment. More often than not, this translates into budget constraints or even reductions.

It is no secret that the biggest share of costs in most IT departments goes to human resources. After all, information technology is an industry characterized by change, and the need to hire and retain staff who are current on the technology is a very expensive prospect. It should therefore come as no surprise that when IT budgets are capped or cut, fixed or shrinking headcounts will be the result.

Another common method being used to reduce costs is data center consolidation. However, if not accompanied by other organizational and operational changes, consolidation can be a costly mistake. Massing more machines into fewer data centers is increasingly seen by organizations as a way to gain greater operating efficiencies. Indeed, centralization of systems and systems management resources may lead to economies of scale, but this is not always the case. If consolidation of systems and SM staff is simply mandated, but is not accompanied by the creation of new processes and tools for central or remote SM, it is unlikely that greater efficiency will be achieved.

Factors Increasing IT Workload

A number of trends are contributing to IT's increased workload. Some of these trends are caused by business changes, and some by technological

changes. Regardless of the root causes, IT must rise to the new challenges while coping with the downsizing pressures already examined.

The first trend is growing pressure from end users for IT to provide increased service levels. Part of the pressure stems from new business processes: in today's more competitive markets, companies must bring products to market faster, which in turn requires engineering, marketing, finance, and manufacturing departments to re-engineer their core processes. The end result is that end users expect more from IT, specifically higher uptime, broader access to data, reduced application response time, faster problem resolution, and increased flexibility. Another contributor to end users' rising expectations is technological: distributed (and especially client/server) computing gives users unprecedented access to data throughout the organization, as well as more control over their personal use of the data. Having tasted the fruits of desktop access to information, users find their appetites growing and expect to be fed a steady diet of data from more and more remote and exotic locales within the company (and often beyond).

Migration from proprietary to open systems is a large factor adding to IT's workload. Because the benefits of open systems derive from vendor independence, IT departments now find they have many, many more vendors to manage than in the years of proprietary systems. While this multivendor proliferation may yield the benefit of favorable price/performance points, it brings with it a multitude of management problems. With more platforms to manage, the systems manager and operators must learn new operating systems and SM tools. It is difficult to hire staff trained in the new technologies and equally hard to keep existing staff current. Because all of these platforms and protocols must coexist in your environment, a host of interoperability headaches usually crop up.

Another technological trend, closely related to and building on open systems migration, is client/server implementation. As organizations implement client/server architectures based on open systems, they experience the advantages of increased vendor choice, better price/performance ratios, and rapid access to information by end users. However, many are finding that the added burden of controlling and managing the distributed LANs, server, and clients is a hidden cost. The complexity of the distributed environment is heightened by interoperability problems, multivendor finger-pointing, decentralized purchasing and change management, and the technological newness and complexity of the architecture. Proliferation of interconnected LANs forces MIS staff to learn new protocols and tools. The distributed nature of the architecture creates many more points of failure. Because end users gain greater access to data, their demands increase. Desktop machines and

distributed servers fall into the expanding realm of the systems manager. The existing larger machines in the data center now take on a new role as servers. Finally, IT has access to very few tools and must use new and undocumented processes to manage client/server environments.

The Net Effect

In summary, then, IT departments must work harder at systems management today. While the fundamental tasks and equipment are largely the same as in the 1980s, business conditions and technological factors are changing the way the equipment is deployed and, in turn, the way SM tasks must be performed. The increased burdens placed on IT by rising end user expectations, open systems migration, and client/server implementations, coupled with the harsh financial realities of doing business in the '90s, have the net effect of squeezing IT's internal resources. The remainder of this paper will examine partnering with an outside service provider to maximize the effectiveness of your internal systems management resources.

Partnering Decision Criteria

The Partnership is a Continuum

There is nothing new or radical about the concept of IT departments' partnering with outside service providers. Now as in the past, very few companies do everything in-house. An example of a very common partnership is the nearly universal practice of purchasing hardware service contracts from the equipment vendor or an independent service organization (ISO). A very few IT departments have historically attempted to provide all maintenance and support via in-house resources and have used outside vendors only for parts and occasional cooperative support.

At the other extreme end of the spectrum lies total outsourcing, often delivered under the auspices of a facilities management (FM) contract. The media hype of the last five years to the contrary, total outsourcing is seen today by many as having fundamental pitfalls. In particular, companies that have signed lengthy (often 10-year), comprehensive FM contracts often cite the loss of control, difficulty of enacting change, and lack of vendor understanding of the client's core business as reasons they later regretted the decision. As a result, many big FM deals have gone sour, and most industry consultants expect fewer FM "mega-deals" to be signed in the future.

Outsourcing in the past has often been viewed as an all-or-nothing prospect. However, in between the extremes of total self-sufficiency and

total outsourcing lies a spectrum of partnering possibilities. The difficult part is deciding which tasks are best done in-house versus by an outside firm.

While the final determination of which tasks to turn over to a service provider is yours, it may be helpful to approach the decision using the following criteria:

- Competitive advantage
- Financial analysis
- Security
- Change

Competitive Advantage

For most effective use of in-house resources, IT departments should be focused on building a competitive advantage for their company's business based on deploying technology wisely. Such tasks as developing or customizing applications are usually best addressed by in-house resources. Nobody knows a company's business expectations from--and changing needs for--technology better than the in-house MIS department. Therefore, those tasks seen as essential to building and sustaining a competitive advantage are best retained in-house, while those daily operational tasks that are routine or cumbersome are good candidates for turning over to an outside service provider.

As a general rule, if a task consists of straightforward procedures that can readily be documented, in-house resources can probably be deployed more effectively elsewhere. Many organizations are finding that such operations management tasks as job scheduling and monitoring, backup and data recovery, disk management, and output spooling are best delegated to a service partner.

Financial Analysis

It makes good business sense to outsource when the service partner can provide a task more efficiently than via in-house resources. Faced with the economics of a make-versus-buy decision, few companies would choose to build their own computing hardware, program their own operating systems, or for that matter, provide their own basic hardware maintenance. However, the range of tasks involved in systems management require a more subtle evaluation of make-versus-buy. Fundamentally, there are only three ways to save money in systems management, namely centralization (economies of scale), automation (routinize repetitive tasks), and elimination of tasks (management by

exception). An outside service provider may already have achieved the critical mass required for operating efficiencies via centralization. It is also possible that an outside service provider may have developed the tools and processes needed to automate and eliminate SM tasks.

It therefore becomes extremely important to ask potential partners to demonstrate how they will achieve costs savings or service improvements. If a service firm's justification is solely that they will implement off-the-shelf SM tools, you may be wise to look elsewhere. For example, you may be able to purchase and use Maestro to automate job scheduling as well as they can.

In performing financial analysis, don't make the mistake of comparing apples and oranges. In order to accurately compare in-house to external costs, it is critical to quantify all internal costs. Include in your analysis such "hidden" costs as users' lost productivity from downtime, part-time ("shadow") departmental LAN administrators, and hiring and training qualified staff. By sharing your in-house cost data with a trusted service partner, they can help quantify these hidden expenses.

It is interesting to note that while costs associated with managing systems in the data center are often well-known and readily quantified, most organizations do not have a cost model for the operations of their distributed computing environment. Why is this? Just as the processing power is shared between the servers (often themselves distributed) and the desktop machines, so the administration of the environment is a distributed function. Often, some of the tasks are performed by central MIS (e.g., LAN monitoring, configuration management, server maintenance and backups), while some are done in the departments (e.g., client software distribution and license management, inventory accounting, initial fault management). Moreover, many of the departmental administrators have other primary jobs to do and are serving in their secondary roles on an ad hoc basis only. By turning over the administration and management of the client/server environment to an outside service company, many organizations can gain greater control and consolidation over all tasks. This often leads to substantial cost savings.

Security and Data Confidentiality

No analysis of which functions to outsource would be complete without thorough consideration of the security of the environment and the confidentiality of the sensitive data on the systems. A common assumption is that using an outside company to provide systems management tasks brings an increased exposure to security problems.

Not necessarily so. The use of an outside service provider, in itself, does not increase risk. Often, it is the methods that will be employed for systems management that are suspect. For instance, if an outside SM provider suggests centralizing a number of systems to economize on management costs, the underlying cause of concern might be that users will require remote access over a network where previously they used dedicated lines. The same concerns would need to be faced if in-house IT chose to consolidate resources. Another concern might be that the SM provider plans to use remote tools to manage the distributed systems: however, remote management will usually save money and may be extremely secure if the right tools and processes are employed. Even moving systems off-site to the service company's premises can be as or more secure than operating them on-site, depending once again on the quality of the vendor's tools and processes.

The bottom line is that using an outside vendor can actually increase the security of operations by employing documented processes and advanced tools. But ask the necessary questions before you choose the service firm to be convinced that all reasonable steps will be taken to ensure security.

Taking Advantage of Change

A concept that is gaining considerable attention today is that of transitional outsourcing. One of the pitfalls of total outsourcing is that once the long-term contract is signed, it becomes very difficult to enact change. For example, suppose that a pharmaceutical company signs a 10-year data center outsourcing contract and, two years later, decides to roll the legacy systems over to a client/server implementation. Under the terms of the FM agreement it would be very hard to roll over the platform, and to avoid penalties the company would very likely have to forego implementing a change that could have given them a significant technological advantage.

Transitional outsourcing avoids this problem by structuring the contract with start and end dates chosen to coincide with a period of change. For example, suppose the same pharmaceutical company decides they wish to roll over their legacy systems to client/server over the next three years. They prudently sign a three-year transitional outsourcing contract. At the end of the contract life, they have a fully implemented client/server environment, understand how to manage the new architecture, and have enjoyed the benefits of a predictable annual cost to enact the change.

Transitional outsourcing can take on one of three forms:

1. Leverage the service firm's expertise to jump-start new technology.

2. Outsource the management of the legacy systems to focus in-house resources on learning the new platforms.
3. A combination of (1) and (2), above.

Ensuring a Win-Win Partnership

Having used the above criteria to decide which SM tasks to turn over to a service partner, the IT department's next job is to set up a partnership that will be productive and beneficial to both parties. This process entails two stages: selecting a partner or partners, and structuring the contract for mutual gain. Fortunately, neither of these tasks is very difficult if the negotiation is handled in a collaborative rather than competitive manner. Bear in mind that you have selected where your company should be on the systems management service partnership continuum; now is the time to implement your plan for maximum leverage to both organizations.

Vendor Selection Criteria

As with entering into any key partnership, the choice of vendor is critical in making the SM service partnership a success. Industry consultants have conducted a number of studies asking IT organizations what criteria are important to them in selecting a systems management firm. The following criteria often emerge as among the most important.

- **Technical Expertise**: Whether an organization plans to outsource the management of their data center, WAN/voice networks, or client/server environment, they must select a partner with a demonstrated command of the requisite skills. In order to get the data to evaluate potential partners, ask the vendor and their references about their tools, people, and processes. If possible, ask to see the facility from which they manage customers' systems. Get specific information regarding the technical environment you plan to entrust to the vendor. For example, a vendor's mainframe FM experience is of no use to you if your organization plans to ask them to manage your client/server environment.

- **Cost**: Everything else being equal, it is important to select a vendor on the basis of price. However, don't cut corners by emphasizing price to the exclusion of the other factors outlined in this section. More important than the dollar amount of the advertised cost savings is an understanding of how the vendor will achieve the cost savings. Remember that the vendor, unlike most in-house IT departments, is operated as a profit center and must cover both overhead and profit margin. So, if the vendor tells you they can save you money, they must demonstrate how they plan to do so while still meeting their own

financial objectives. Similarly, beware of "lowball" bids: if a vendor bids a much lower cost than others, find out why. Perhaps the level of service will not be as high or the other terms of the contract are more restrictive.

- **Size and Financial Stability**: The selected service partner must have the financial resources to retain the working capital required to meet your needs. A track record of financial stability (and the bond rating to demonstrate it) is helpful. Particularly if you plan to sell some of your capital assets to the service provider, they must be fiscally sound.

- **Geographic Coverage**: Your partner must be where you are in order to manage your systems. Even if remote management will be used for your distributed systems, what happens when on-site service is required? A potential partner should be able to show you the process they will use to provide the service levels guaranteed in the contract.

- **Working Relationship**: Bearing in mind that you are selecting a key partner to whom you will entrust the management of your critical systems, it is important that the partner be easy to work with. Have your past dealings with the vendor been satisfactory? Have they demonstrated a commitment to quality? Do you have the mutual trust to ensure that contract or performance concerns will be handled fairly?

Contract Issues

Having selected the vendor, it is now necessary to develop the terms of the contract. While each vendor will have a standard set of terms and conditions, your systems management needs will necessarily differ from those of other organizations, so you should insist on enough flexibility from the vendor to meet your key needs. The following are a few important points to consider before signing any outsourcing contract.

- **Clearly Specify Required Service Levels**: Any systems management service contract should include a service level agreement (SLA) which clearly specifies the level of service the vendor must provide in order to meet your business needs. Be precise in defining such terms as "response time" (which two events define the start and end of the period being measured?) and "uptime" (of what?, measured how?).

- **Insist on Periodic Reporting to Verify Performance**: The service firm should provide you with periodic (perhaps monthly) reports measuring

the performance of the systems, networks, and applications being managed. The reports should be tailored to your organization's needs and should specifically measure performance in terms defined in the SLA.

- **Specify Penalty/Escape Clauses for Non-Performance**: No contract is better than its enforceability, so spell out what the penalties will be if the service provider cannot meet the terms of the SLA. You will already have selected a partner you trust, so most performance concerns can be handled informally; however, if a problem is major and persistent, these clauses will give you important leverage.

- **Choose a Contract Duration that Meets Both Parties' Needs**: Your partner needs to turn a profit. You desire maximum flexibility. Because systems management is a partnership, both parties must work together to choose terms acceptable to both. This is especially important in the area of contract duration, because too long a contract life may erode the client's ability to remain responsive, while too short a life will not allow the vendor to make a profit. Increasingly, outsourcing contracts with two-year to five-year durations are being structured as mutually satisfactory.

- **Allow Flexibility to Change Contract when Your Business and/or Technical Needs Change**: Closely related to the last point, the contract should be flexible enough to allow your organization to take advantage of new technologies or processes. In fact, one of the primary advantages of partnering with an outside service provider is that your partner can help you stay current with new technologies. Indeed, a contract can be structured so as to have the service partner roll out new technologies for the client on an ongoing basis and for a fixed price, thus ensuring that you will always have access to the latest technology.

In Closing

Just as every organization's systems management needs are unique, there can be no single SM solution that is right for everyone. This paper has argued that few companies could or should try to manage every aspect of their IT environment with just in-house resources. Similarly, most organizations would be wise not to outsource every aspect of their systems management in a long-term, FM-type contract. In between these extremes is a spectrum, and only you know where on the spectrum lies your optimal balance between in-house and external resources.

We observed a number of trends taking place over the last decade, the net effect being to squeeze most organizations' in-house IT resources, often to near the breaking point. Partnering with an outside service firm to manage some of your environment lets you do more with less while retaining full control over your systems. The methods outlined here have been helpful to organizations looking to partner with service firms for maximum leverage of their internal resources.

An interesting transformation has occurred since the 1970s and 1980s. In the past, the most strategic decision an IT manager had to make was the choice of a hardware vendor. As important as the choice of hardware remains today, it is equally strategic to choose your SM service partner with care, as you will be working with them daily.

The Realities and Myths of Client Server Application development in the '90s

David W. Haberman
140 Wood Road
Braintree, MA 02184
(617) 356-8710

Before we start any discussion of technologies it is important to understand why we implement technologies in a business environment. There is one reason and one reason only that we implement technologies in a business environment. The reason is to make money. We are in business to make money. We make money by increasing revenue, saving money, and remaining competitive. If the technology does not meet one of these criteria then the reality is there is no reason to implement it.

Keeping this in mind let me relate to you how I got interested in this topic. About a year ago I met with the MIS director of a good sized organization. When I asked him what his plans were, he said he was going with client server. When I pressed him as to what that meant he went on about his asymmetric 486 processors, Novell network, and WAN. I then asked what software he was going to run. His answer was "I don't know, Can you recommend something". I immediately went ballistic(on the inside). If I remember correctly business need drove the application, application drove the hardware. we covered this in MIS 101. I then decided to do all the research I could on client server and how it fits into the MIS environment, the results were this paper.

What I hope to cover here is:

What is client server

Why is client server

What is needed for client server

Where is client server today

How do we prepare for the future

What is client server

If you search the industry publications and the industry vendors you will find that everyone has their definition of client server. Instead of asking the vendors Computerworld published surveyed their readers to get a definition of client server. This definition was published in the 12/17/90 issue of Computerworld.

Client server was defined as :

The client end and the server end can be distinguished from each other but interact seemlessly.

The client portion and the server portion can operate on different platforms but don't have to.

either the client hardware or the server hardware can be upgraded without upgrading the other platform.

The client server system includes networking capabilities

A shorter definition might be:

The ability to process data across platforms and/or CPUs regardless of where the data or the process resides.

The term client server is typically applied to the ability to integrate PCs or workstations into the mini-computer or mainframe environment. The typical environment uses a server as a kind of traffic cop for data and programs while the client processes the programs and the data. If your apply this environment to the definitions above you see that this is only one facet of true client server. In fact if you use the definitions above you will find that the terms distributed processing and cooperative processing fit the description better than the term client server.

There are many different client server strategies. However when you look at the definitions above you notice that client server has very little to do with hardware. The true definition of client server i between applications. The hardware is transparent in the definition.

Why is client server

To understand the importance of client server in today's MIS environment you must understand the make up of the corporate data processing environment. In the early days of computers organizational structure was top down. Since the business was managed centrally, the data management for the business was centralized. In the Seventies organizations realized that the centralized organization was having trouble responding to changing business needs. Organizations started to break off and decentralize business units. As these units decentralized we also decentralized our data processing. Each business unit bought and managed their own data processing systems.

In the 80s we realized that with all this decentralization organizations had trouble responding to the changes in business because. Decentralized businesses did not have the ability to share information for use in corporate decisions. The need for centralized access to decentralized data become apparcnt.

Along with this need came enabling technologies. The advent of PCs put the power of a computer on every desk top. Networks were developed that allowed PCs to share data and programs. Application standards such as MS Windows, SQL, and MOTIF were developed. These technologies allowed users of disappoint systems to start to share information.

Along with these technical factors were business factors. PCs provided a great deal of p[processing power at a very low cost. Disk storage on a PC cost significantly less than mainframe storage. Graphical user interfaces made computers easy to use.

PCs also provided powerful easy to use development environments. In fact with tools such as dBase, Paradox, and Lotus developers could deliver full applications faster than any mainframe environment. Because of all these factors the concept of client server has gained great popularity.

What is needed for client server

To create a client server environment we need five factors. They are:

Computers that are physically networked together

Operating systems that communicate with one another

Inter networking devices

More than one communication protocol

Application development tools.

Looking at these five factors where does client server stand today.

Physical networks

Today we have the technology to link any system to any other system. Network adapter boards are commodities and can be purchased in boxed sets at any computer store. Hubs are available to tie different types of network boards together. The cost of the physical network has dropped significantly as the technology becomes easily available.

Inter networking devices and Communication protocols

Communications protocols exist to allow any system on the network to transfer data to any other system on the network. The problem is that each system has its own communication protocol. SNA, TPC\IP and OSI are just a few of these protocols. Bridges routers and gateways filter data and selectively forward it as defined by different protocols.

Operating systems

This is a technology that is just emerging. Standards for operating systems are just now beginning to be accepted. UNIX and POSIX are two of the driving standards in operating systems but they are by no means universally accepted. In fact even these "standards" come in different versions.

To give you an example of the importance of operating systems in the client server environment lets look at file names. In the MS DOS environment a file is described with 8 characters and a three letter extension. In UNIX the standard is 255 characters. In MPE the standard is 16 characters in a file.group. account structure. It is impossible for an MPE program to call a 255 character file name and process it under MPE. If this is just one example think of all the other issues involved in different operating systems.

DCE the Distributed Computing Environment from the Open systems Foundation was designed to manage the multiple operating systems in a distributed processing environment. DCE provides a uniform set of services available from anywhere on the network allowing applications to access programs and resources anywhere on the network.
DCE handles:

Remote Procedure Calls (RPC)

Directory Services

Time Services

Security Services

Threads Services

Remote procedure calls include an interface definition language. This language is a set of C like structures that allow the developer to translate data representations from one platform to another.

The directory services acts as a [post office for applications and data. The directory services store the names and addresses of programs and data on the network and direct call from one area to the correct address.

The time function manages the synchronization of different system clocks on the network. Security encodes and decodes network traffic to protect the data as it travels through the network. Threads allows a single program to run concurrent processes.

Application Development tools

Application development tools are made up of four pieces.

Data Bases

API (Application Programming Interfaces)

End User Query Tools

Programming languages

Today most application development tools are available for developing the client end. These tools do not allow the development of complex logic on the server platform. These tools do not allow you to track the impact of changes to the program on the entire system.

Most of the application tools that allow you to develop distributed systems are based on relational database management systems using SQL. Most of the tools that develop both client and server pieces are database dependent.

In a recent computer world survey 69% of Fortune 1000 companies claim to have some type of server database. The majority of these systems are for end user computing or executive reporting. It should be made clear that this 60% means that these organizations have at least one client server application and does not mean that these organizations have committed to client server.

In fact as of March 1992 only 29% of all companies surveyed by Computerworld were planning to move, moving or had moved to client server. The majority of these client server systems were some type of end user reporting.

Barriers to Client Server

J.C. Pollack Associates did a survey to discover the barriers to client server. The discovered that Networking issues, hardware issues, training, and convincing IS are not the major barriers to client server. The two biggest barriers to client server implementation are cost of implementation and management buy in. In fact the two are closely related.

There is a great misconception that client server costs less than any other type of application development. This is not true. Most comparisons of client server application development are done against mainframe shops. Mainframe application shops typically have a lot of overhead and systems programmers. The rules that apply to mainframe shops do not apply to minicomputer shops.

Studies have also shown that the cost of network management, training, and support are rarely added in. Nolan and Nolan did a study of the cost of end user computing. They asked their clients how much they spent annually on end user computing. The typical answer was between $2,000 to $6,000. When Nolan and Nolan actually audited the sites they found that the typical annual spending on end user computing was between $6,000 to $20,000. Most of these costs were hidden costs of lower productivity and training.

Development costs on a client server environment are also typically compared to Mainframe shops. PC tools are far easier to use than mainframe tools. However the tools in the minicomputer environment are equal to, and in some ways superior to PC tools.

Preparing for the future

In looking at the directions in development we realize that hardware is becoming a commodity. Operating systems and database access are becoming common. The biggest investment we will make is in application development. We need a tool that will meet the needs of today's developers while remaining flexible for the future. Platform, database, operating system should be immaterial to the organization. developers should choose a tool that lets them meet today's needs and prepare for the future.

Paper No: 6005

Thinking Relational

Alan Camburn
Richard Irwin Associates Ltd., Inc.
373 First Street, Suite 100
Los Altos
CA. 94022

Tel: (415) 949-0180, Fax: (415) 949-1293

Introduction

To achieve and maintain any competitive advantage in today's marketplace, organizations are recognizing the need to ensure their Information Systems (IS) are in better shape than in the past. Strategic Information Systems Planning (SISP) a means by which the business objectives of the corporation are closely tied into IS planning and development is becoming essential to ensure optimum use of the organization's investment in technology. The data driven approach is being recognized as an essential technique to cement the centralized planning and modeling of the organization's data. This modeling of the data ensures that the organization's business has been understood and accurately reflects its information requirements. Having achieved this goal it is essential that we are then able to manage the data effectively.

The management of information is becoming an increasingly complex task for many organizations; being able to access the data quickly and efficiently relies on good design. Increasing degrees of functionality within the assorted Database Management Systems (DBMS) presents us with a situation by which the design of a business transaction must consider a few more variables than we have been used to.

If we look at the evolution of information management over the years, it is interesting that the significant changes from one generation to another have not only come about because of the technology improvements, but also because of the recognition that we need better ways of managing our data. Each subsequent generation hasn't just been a collection of bits but an attempt by the assorted software vendors at enhancing the underlying software to complement the real world data requirements of an organization.

Relational technology is now with us; there are a number of products available within the Hewlett-Packard environment, and many sites are starting to use the new technology. Tried and tested methods gained working with TurboIMAGE are no longer applicable, a shift in thinking is required to ensure success. We must start thinking SQL and not just map our TurboIMAGE habits into SQL. The key to this success is the clear understanding and definition of the business transaction. How well this is done will directly influence the following :

Performance - Good index design comes about as a result of diligent access path analysis, but what about normalization?

Integrity - Constraints on your data can be imposed in a variety of ways within the RDBMS, how does this affect your program design?

Use of 4/GLs - Productive, almost certainly, but must we make any compromises for the more complex business transactions?

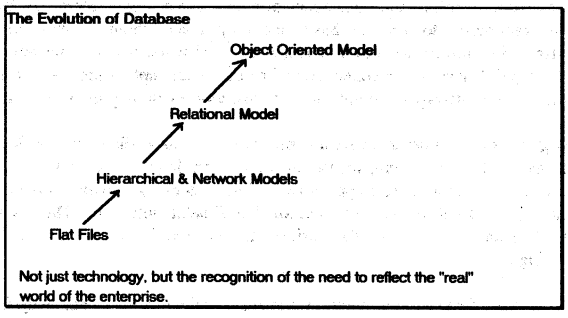
Data Distribution - With open systems becoming a reality and client/server configurations a practical necessity for some, the database designer must ensure sound transaction design to maintain the integrity of the data.

This paper addresses the issues concerned with the design of business transactions within a relational environment, and suggests some ideas in laying the foundation for future developments.

Information Management

Commercial application systems of old used conventional flat files and in general were batch driven systems, which happily satisfied the time consuming repetitive tasks for which they were designed. The concept of shared data very much in its infancy in the early days was not perceived as an issue. However as organizations started relying more and more on their data, and the way in which they both wanted to access and use their data, changed. It was quickly noted by many that the flat file batch oriented approach was somewhat constraining; ie. multi-user access was difficult, and the locking and security capabilities were limited. Regardless of how well the data was structured within the files (this typically was rare anyway) the business rules surrounding the data had to be embedded within program code. What is meant by business rules, is essentially the interdependencies within the data, ie. it is not possible to enter a foreign exchange transaction unless valid currency codes are used.

The arrival of the hierarchical and network models was a clear drive to address the problem. The database philosophy, ie. that of sharing data and structured to some degree to fulfill the basic data requirements of an organization, essentially drove the need for these models. Thus the DBMS was born aimed at creating an environment where data could be shared across multiple business transactions, subjected to consistent locking and security provisions and holding some business rules within the underlying software. This ensured that all accessors were subject to the same integrity constraints inherent within the database. Software built to support these models has been and still is very successful. TurboIMAGE, a well known and loved DBMS is very robust, performs well and has a strong user base.



Within the hierarchical and network DBMS' we are constrained by their rigid structures. We must consider these structures when mapping our logical data model to the physical, and make allowances for them. These structures are a way of implementing simple integrity constraints or business rules, but assume that the inter-relationships within the data are always on a parent-to-child basis. When this is not the case we have to devise a work around. On mapping to the physical we have to decide what type of "dataset" we must best use to support the data structure defined in our logical model.

The relational model on the other hand does not constrain us by chain heads and pointers. All relationships are supported naturally within the data, as are our integrity constraints. There is no concept of a child being allowed only one parent(hierarchical) or multiple(network). This means essentially that all entities identified at the logical level become tables - equality for all! Out of Codd's original 12 rules one can identify two very fundamental necessities for any shared database structure :- the structure of the data itself, ie. how it is organized, and integrity constraints.

The next generation in the evolution of information management is Object Oriented Design; still very much in its infancy, but getting closer to an ideal. Not just another collection of bits, the concept is that of representing data, relationships and integrity constraints upon that data (relational), but also permit the encapsulation of the business rules that operate on that data. This gives us more than just software but a means by which we can truly represent the organization's data and processing requirements at the DBMS level. All accessors derive the same meaning and consistency is maintained across not only a single application but across multiple.

Being in its infancy there is not really any suitable software around that can be successfully applied within a multi-user commercial application environment. In addition, the whole concept has not really stabilized yet, and no clear definitions of the components of Object Oriented Database design have emerged.

Relational DBMS (RDBMS) have now matured at an accelerated rate over the last three years. A common database language (SQL - Structured Query Language) across multiple products, and the drive towards the open systems approach for those with multi vendor networks, has meant that an increasing number of organizations are now considering the move to a relational platform. In making the move to relational and make optimum use of the technology it is essential to :

* Understand the fundamentals of the relational model
* Model the business data requirements and in so doing identify the business rules
* Develop a technical architecture and choose how to use each component of the new technology(s); in particular choose how to enforce the business rules
* Design systems that tie back to the business model, and that make consistent use of the technical architecture

Once the logical-to-physical mapping of relations to tables is done, the success of a system depends on understanding the business transaction in the relational environment.

Business Transactions

A business transaction can be defined as a logical unit of work that must continue to completion to ensure consistency and integrity; it's either all or nothing. The most quoted example is where a bank wishes to transfer a sum of money from one account into another. Clearly the whole operation must be completed otherwise there is the danger of having a sum of money floating around looking for a home! With RDBMS, because of their inherent characteristics, we must seriously examine more carefully the design of our business transactions, if we don't we risk compromises to the following :

* Integrity of the business rules
* Data consistency
* Optimum concurrency of access

To meet these three requirements we must have clearly identified the business processes associated with the business area under study, and ensured that at a conceptual level the data model supports the required processing. To get to this stage means that we must have clearly defined the information requirements of the given business area, without which we cannot be sure that the resulting system will :- 1)satisfy the business objectives, 2) meet the critical success factors via system objectives.

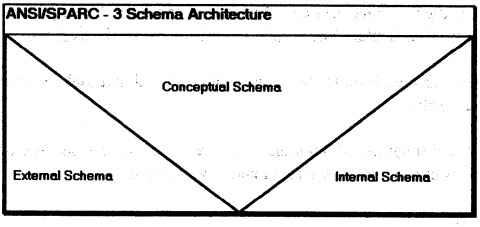
With this clear definition of the business's data and functional requirements we are better equipped to enter the business transaction design phase of the project. It must and will have an impact on the following :

* Use of 4/GLs
* Integrity
* Performance
* Data Distribution

Use of 4th Generation Languages

The use of 4/GLs is on the increase as is their overall quality. Clearly aimed at improving productivity and ease of coding. In much the same that SQL provides us with a higher level language than the DBintrinsics for accessing our data, 4GLs have come up a generation from standard 3/GL. This is fine, but we must pay for this convenience somehow, and experience has shown that it is already hard enough to build a system accessing a RDBMS utilizing a 3/GL.

Most 4/GLs available now for accessing RDBMS are being designed to run on multiple platforms and access multiple file systems as well. In defining their respective dictionaries some products claim adherence to the ANSI/SPARC - 3 Schema Architecture



The 3 Schema Architecture essentially embodies an ideal where IS development is concerned, its philosophy is that an application can be defined in terms of the data it processes :-

Conceptual Schema - The core of the architecture, it contains a complete definition of the data structures as established in the data modeling phases of the project and provides the stability for all future developments.

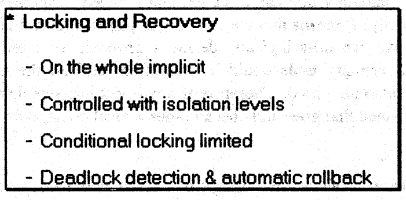
External Schema - Within this area are created the users' views of the data. It is where the business transactions are defined dependent on the functional requirements of the system. Additionally it is where the common "look & feel" of the system is implemented in terms of the graphical user interfaces (GUI), which could be PC based and windows driven.

Internal Schema - This area provides the link to the physical database out on disc. We should be able to change the platform without necessarily having to alter either the data structures in the Conceptual Schema or the business processes in the External Schema.

The functionality in which the external schemas can be defined will vary from product to product; with some more flexible than others. In designing the external schemas, irrespective of how complex they are, we should not have to compromise the data structures held within the conceptual schema in order to work around any shortfalls within the 4/GL. Performance is a separate issue and is addressed later within the scope of this paper.

The method used to display data retrieved from multiple tables on the same screen will vary from product to product and some are more "user-friendly" than others. In the worst possible cases compromises to the data structures are required, ie. de-normalization. This may be due to the fact that if the 4/GL works on the premise that one screen is essentially one table in your database, it may require the user to "trigger" another screen to display the data from table 2, and so on. This is not necessarily a problem but does require more work on behalf of the end-user. With the more complex business transactions that access the more volatile tables in the database we must decide whether the cost of providing the end-user with something more amenable outweighs the cost of compromising the data structures. This decision can't be made purely on technical grounds but must take the critical success factors of the system into consideration as well.

Most 4/GLs can be run on multiple file systems, therefore data retrieval capabilities of the products assume that a query could access data from different file systems within the same transaction. This means that what could be a multi-table SELECT statement in the SQL language is decomposed into the relevant number of single table SELECT statements. It has been proved that a well constructed multi-table join performs faster than the equivalent number of single table SELECTS. The work around for this is to construct a View within the RDBMS and reference it in the 4/GL. To build these sometimes complex Views requires programmers to understand not only the data structures, but also the SQL language, so that the appropriate "manual" navigation around the data structures can be achieved.



Another area that requires careful thought and design is that of locking and recovery. Locking capabilities vary from product to product and are quite different from the straightforward and effective locking capabilities within TurboIMAGE. The ANSI/ISO standard is attempting to provide some guidelines in this area, but when it comes to locking at either base, table, page or row level this tends to be "implementor defined"; all the SQL standard can do is specify what are known as isolation levels.

An isolation level is essentially specifying at the start of the transaction the granularity of locks that are applied on your behalf throughout the duration of the business transaction. They specify whether you wish the transaction to read only, and if so whether it is a "dirty" read or committed data only. With regards to write operations we have to consider the optimum level that will best suit concurrency of access by other transactions.

It is not the intention of this paper to examine them individually, but there is no doubt that it is essential that the correct blend of isolation levels is arrived at dependent upon the particular requirements of the environment.

Integrity

Integrity constraints at the physical level address basic table integrity, and referential integrity. Table integrity addresses the basic requirement of uniqueness, ie. the values contained in the column customer-code for the table Customer are unique. Referential integrity is the situation where the values contained in the column customer-code in the table Order are for valid customers; any violation of either of these two results in an error condition.

The way in which the various products implement this concept varies, but it is addressed in the ANSI/ISO SQL standard, and the products are implementing the standard as defined.

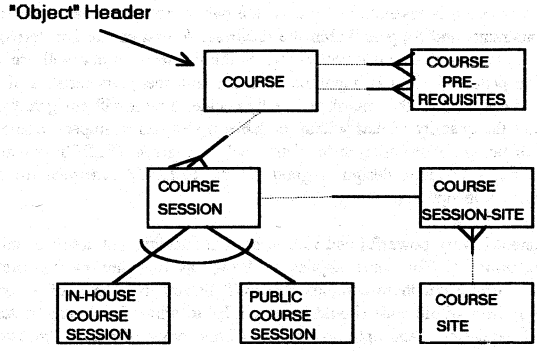
This presents us with a problem when using 4/GLs. Typically they all have their own dictionary in which the same constraints can be specified. So the question commonly asked : "Where shall I put my integrity constraints, in the dictionary or in the database, or both ?" If the 4/GL insists that they reside within the dictionary and they also reside within the database, is the 4/GL intelligent enough to bypass one or the other?

Ideally they should reside within the database; as it means that any accessor regardless of mode of access, will have the same constraints applied. It also means that if the database is to be ported to another platform, which may have a different mode of access, the same rules apply; thus consistency is maintained. Clearly the appropriate decision has to be made within your environment and considerations such as the portability of the database does not necessarily apply to all.

In addition to the basic integrity constraints and column constraints, we can now specify triggers and procedures that can be stored within the database itself. A trigger is executed when something happens to a table, and a procedure is something that has to be explicitly called. This now broadens our horizons in terms of the concept of integrity. It means that we can now implement more complex business rules within the database. For instance we can now logically define a grouping of inter-related tables as being an important data object, ie. a primary table could be identified, and all the other tables that essentially describe that data object are also identified. Therefore if any event happens against any of the subordinate tables a trigger could be activated that goes and checks other logical constraints that could not otherwise be checked.

The following example illustrates where the concept of objects could be used. It is a part of a training administration system and the entities represent the necessary information that the business needs to hold with regards to Courses and Course Sessions.

The relationships between the entities represent the business rules inherent in the business. For instance, an occurrence of the entity type Course may or may not have one or more pre-requisites, and that particular course may be the pre-requisite of another course. A course may, over a period of time, be presented at one or more Course Sessions, and a Course Session must be either an In-House Course or it must be a Public Course, and so on.



In one scenario and what could be for business reasons, the entity Course has been selected as the pivotal entity in the group. That is to say that all the subordinate entities effectively describe an occurrence of a Course. What this means is that when it is necessary to add additional Course Sessions into the system, together with the type of session (In-House or Public) and possibly site information, that information is completely dependent on there being an occurrence of the relevant course. Updates to any part of the structure would have a rippling effect and would need to be checked to ensure that the consistency of the business data is maintained.

Basic referential integrity within the DBMS will suffice between pairs of entities. However with the "cascade" option becoming available, the decision on how to implement integrity constraints becomes more complex. The cascade option gives the ability of being able to specify at create table time, that when we delete an occurrence of Course, all its related Course Sessions should also be deleted; alternatively we may decide to restrict that delete occurring. If there is a sound reason for deleting all course related information, do we then wish to delete all the Course Sites related to that course? Probably not; as we may wish to host other courses at those sites. Therefore dependent on what the business rules are the overall integrity of the above structure could become reasonably complex and not easily enforced within the DBMS. As a result you would need to enforce those rules in another way to maintain the flexibility required by the business.

On further analysis it may be decided to create three objects from the above structure. Still maintain Course, but nominate Course Session as an object in its own right as well as Course Site. Course Session - Site would probably reside in the object Course Session. This probably makes more sense but complicates the business transactions; as they now have to cope with crossing multiple object boundaries. Cascade updates and deletes must now be even more strictly controlled and would probably not be implemented using the feature within the DBMS itself. The more likely outcome would be to use triggers or procedures as they provide the most flexibility and so can better cope with the exceptions.

Performance

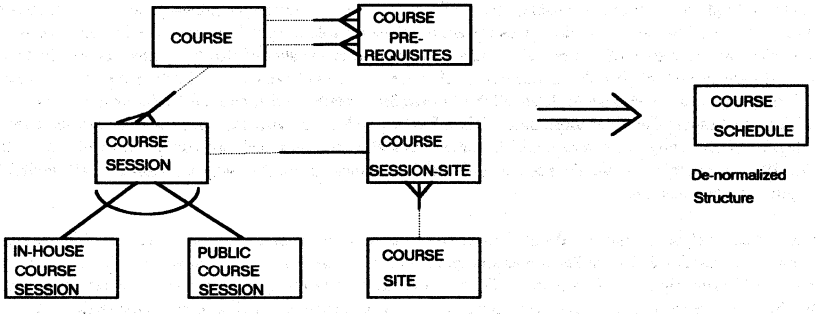
Good performance is a difficult thing to quantify. Different users of the system will have different requirements. Despite this, a good technical design, a good database design and volume testing at the appropriate stage can go a long way to avoiding some of the pitfalls normally associated with RDBMS.

When building the database it is essential that transaction path analysis is done to ensure that there are no surplus indexes, constraints and triggers within the database. A first-cut design derived from the logical data model is a good place to start, but it will evolve as the transactions that will use it are built. When exercising transaction path analysis it is important to understand the characteristics of complex SELECT statements. This is no easy task as the optimizor usually decides how it will navigate the data and whether to use indexes or not; the quantity of data within the table(s) also has an impact. Some RDBMS products give us the luxury of being able to analyze the "run-tree" of a given SELECT statement. This can then verify or otherwise that:- 1) index design is good, 2) the SELECT statement itself is sound, 3) the clustering of the data is at its optimum.

The SELECT statement is very powerful and rich in its functionality, but abuse of this functionality can cost dearly in performance. The same applies to Views as they are merely pre-defined SELECT statements anyway. Testing out these complex SELECT statements as part of a quality control phase prior to incorporating them within code should become a habit when developing systems. This will then ensure that the database designer can optimize the use of and types of indexes by reviewing the critical paths to the data.

Normalized data structures give us a balance of performance across the whole system. Normalization benefits updates, but does impact the performance of retrievals. De-normalization speeds up retrievals, as does adding indexes, but slows down and complicates updates. Each situation must be examined in the light of the critical success factors of the system. The de-normalization of the data structures should only be considered as a last resort if resilience to any future change to the structures is to be maintained.

Materialized Views



Materialized views give us the necessary flexibility; as we can still maintain the data structures as they should be. A materialized view is the evolution of a SELECT statement into what is a physical table in the database, ie. we have taken a view, created a table in its image and populated it with data from all the underlying tables originally specified in the VIEW.

When I now wish to retrieve all the relevant data to produce a Course Schedule, we need only access this new derived table. Clearly performance is improved as we only have to access one instead of what was previously seven tables, but updates will be affected; as we must ensure the data in the extra table is also maintained. It is a form of de-normalization, but still maintaining the original data structures.

Data Distribution

The whole area of data distribution is probably one of the most difficult areas to resolve. Dependent upon the particular environment, decisions always need to be made over whether to replicate data in remote locations or to maintain a central location. The volumes of data transfer, timeliness and cost of the lines usually determines the resolution.

One of the most common problems is that of a business transaction spanning more than one database each of which could reside on different servers. The two-phase commit feature only now appearing within assorted RDBMS is at the heart of the problem. To ensure that when a business transaction is completed all the associated write operations are made to the databases, the software has to co-ordinate the commit to the relevant servers. Some of the RDBMS products have only recently been able to do this, and TurboIMAGE itself has only recently supported this feature.

Now the two-phase commit is reality, it does put additional pressure on the design of those applications that want to make use of the feature. Everything already mentioned in this paper is impacted in some way and therefore the concept of a sound technical architecture takes on more meaning. Within the concept of a client-server environment careful thought must be given to how the data structures are going to be physically mapped. Decisions on how this can be done, cannot be made until a good profile of the business transactions has been derived, outlining their access patterns, volumetrics, end-user requirements and the critical success factors of the systems being developed.

Conclusion

As our requirements for more functionality within the information management arena increase, so will the technology improve to assist us. The assorted standards bodies are already looking at clearer guidelines within the context of transaction management. To achieve a true open systems environment any combination of products selected for future development must satisfy, or will satisfy the following three main areas that are required:

* Portability
* Interoperability
* Scalability

Many organizations are now recognizing the importance of a sound and stable platform of data upon which to build their business transactions. The design of systems must tie back to the Business data model and make consistent use of the technical architecture. The technical architecture should define how each component of the new technologies are going to be used, such as how and where to enforce business rules. The success of a system will depend on understanding the business transaction and how it can best be implemented within a relational environment.

Paper 6006

DSIS- Promoting Information Standards For Open System Services

Bill Bowman
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, California 94087
415-691 5678

BACKGROUND

The computer industry is seeing a shift to multivendor distributed computing environments. These environments are characterized by networks (LANs and WANs) connecting a range of systems. Increasingly network operating systems are managing not only shared resources but also distributed applications (i.e., client/server). As these systems are deployed throughout customer environments their support and management are becoming strategic issues. Customers are finding centralized proprietary systems have a real advantage in areas of service and management capabilities compared to the new open system environment. Support and system management in this new environment is made difficult for several reasons, including:

1. Much greater intelligence and complexity are distributed to users' desktops.

2. The network assumes a critical role as applications are distributed.

3. Customers exercise the promise of open systems by purchasing multivendor equipment.

4. In their rush to provide functionality and interoperability, vendors have been slow to provide multivendor support and management solutions.

5. End users have limited knowledge and products have not been designed with support and management as a priority. If something doesn't work, the user doesn't know whether it is an application, platform, network or ...

6. Customer IT departments are under pressure to reduce costs while having to support both the legacy environments as well as the new environments. In addition

they often lack the new skills required for implementing and managing the complex distributed environments.

An underlying problem in developing support and management applications for the distributed multivendor environment is the inconsistency of information i.e., what information is available and what it means. An alliance, called the Distributed Support Information Standards Group (DSIS), has been formed to promote the standardization and deployment of standard support and management information across multivendor environments which include PCs to mainframes.

DSIS was founded August 4, 1991 by an initiative from Bell Atlantic Business Systems Services. Membership today includes Hewlett Packard, Microsoft, IBM, Digital Equipment, Sun Microsystems, EDS, Olivetti, ICL PLC, HaL Computer, Tandem Computer, Bell Atlantic Business Systems, Unisys Corporation, Proteon Inc., Landmark Systems, and October Technologies. DSIS is the first consortium specifically aimed at the adoption of standards for support and system management information. The scope of DSIS does not include standard network information since this area has received considerable standards attention from the Internet Engineering Task Force (IETF).

The DSIS consortium does not generate standards. Rather DSIS develops information requirements from the standpoint of support providers. The specification is in a protocol neutral form which can be translated to specific management protocol environments. The specification is given to existing standards groups developing standards for system and network management.

BENEFITS

Today all major vendors support networking standards and are starting to support standard application interfaces like POSIX. Because of the lack of information standards, however, each vendor has a proprietary system management solution(s). This prevents multivendor environments from being effectively supported by a set of integrated management solutions. Multiple support environments increase downtime and lead to higher support costs.

Groups benefiting from the adoption of standard support information will include system end users, system managers/operators, system vendors, and third party maintainers. A key DSIS benefit is the ability to leverage support expertise due to the availability of consistent information from different systems. Consistent information will increase the automation of support tasks and enable more remote problem isolation by having ready access to information about the customers' environment and problem scenario. For example, if a user is having a print

problem, using DSIS information the administrator can determine if there is a printer problem, e.g., off-line, or if there is a print queue problem , e.g., disabled queue.

DSIS information can be used to proactively monitor system resources and prevent problems from occurring. For example a tape drive may report an increasing number of write faults indicating the need to clean the heads. A preventative maintenance call can be scheduled saving unscheduled downtime for the customer and making the support provider more productive.

DSIS includes information useful in helping users get maximum benefits from their system resources. As an example, performance may be reduced because of a shortage of available memory. A performance management application could read DSIS performance information and off load applications to a different system. Software and hardware inventory management is a major challenge with distributed computing environments. Using DSIS information, system managers could regularly generate current inventory information enabling them to better manage computing resources.

Information standards will encourage the development of standard support and management applications for multivendor environments. Recently several companies have been formed specifically targeting this business area. Without information standards, these management applications must support disparate vendor specific information access methods. With information standards, application developers can focus on building applications to meet customers' pressing system management concerns rather than understanding the specifics of each system. Their task of maintaining these applications will be greatly reduced because the system vendors would provide DSIS information as a standard part of new releases.

MANAGEMENT MODEL

To understand the work of DSIS it is useful to introduce a management system model. (See Figure 1- Management Overview). The management platform has the user interface and the management applications. These applications use the standard DSIS information from the managed systems SW, HW and NW components. Components which can return management information are called managed objects. The MIB, or management information base, contains information about the system's SW, HW and NW managed objects in a standard form. The MIB is like a "virtual data base" which resides on the managed system. A management system accesses the MIB data using a standard management protocol such as SNMP (Simple Network Management Protocol).

DSIS information about managed objects are called attributes, e.g., the type of printer interface. Managed objects require object agents which are responsible for getting the information, e.g., this printer has a serial interface, and returning it in a standard form. These object agents are system specific routines. Each attribute has a universally unique registered object identifier which is part of the management request. Therefore the management application can issue the same request to different multivendor systems knowing it has asked for the same information from all systems. Not all information has to be explicitly requested by the management system. DSIS supports the concept of alarms by which the managed systems can report a failure such as "printer out of paper" without having to be polled. Popular management protocols, e.g., SNMP, support alarms. DSIS Phase 1 Requirements includes "read only" information and does not include "actions or sets".

DEVELOPING THE DSIS SPECIFICATION

The DSIS specification was developed over a one year period using the combined resources of member companies. The first step was for each member to go back to their support organizations and ask them for what kinds of information was most frequently needed to resolve support problems. These organizations were primarily telephone response groups doing remote phone-in support. The responses were grouped by area, e.g., SPU, Operating Systems, Printers, etc. In order to better understand and communicate how the information at a system level was organized, a graphical concept was used called Entity Relationship Diagrams (ERDs). Using ERDs all objects and attributes were linked to the central root "system " using various associations. As an example, Figure 2- Printer ERD, shows an ERD for a printer.

At a detail level, however, ERDs become awkward to maintain. This was especially true while the specification was undergoing changes and there was no single graphical package used by all members. We also tried expressing the specification in ASN.1 format. ASN.1 stands for Abstract Syntax Specification Revision 1 which is an OSI standard for exchanging data structures in a machine independent manner. This also proved unsatisfactory because we now were focusing on how the information was represented and not what the information should be. Finally we adopted a concise form which shows the DSIS attribute, its data type, e.g., integer, counter, display string, whether it is a single item or a repeat (i.e., table), and its compliance level. An example of a repeat is a spool queue can have multiple requests and each request has unique information about that request. A single item would be an indicator if that queue was enabled or disabled. An important DSIS goal is to develop "protocol" independent definitions and the concise form enables us to do this. An example of the DSIS concise form

for a printer is shown in Figure 3- Printer Attributes.

Early on we recognized the importance of having guidelines for selecting what information should be part of the DSIS specification. Ideas for guidelines came from the years of experience in developing and using standard network information. Below are some of the key selection criteria for DSIS information:

1. Objects should be as common as possible to ensure their wide-spread use.

2. Avoid management protocol dependent definitions. DSIS information is protocol independent in order to be deployed in different management environments ,e.g., IETF TCP/IP, OSI CMIS/CMIP.

3. Avoid defining objects which impose burdens on critical sections of managed systems or require large amounts of storage. After the objects have been instrumented, the managed system must still be able to function in its primary role.

4. Remember the 80/20 rule: 20% of the information will be used 80% of the time. Too much information is as much of a problem as too little information.

5. Avoid attributes which can easily be derived from other attributes.

DSIS PHASE 1 SPECIFICATION

DSIS phase 1 focuses on vendor independent system information which is common from PCs to mainframes and is important for the support and management of these systems. Phase 1 contains "read only" information about HW and SW components. Future DSIS requirements will address "write" information necessary for robust system management applications. The DSIS information is divided into component groups. Phase 1 includes the below groups:

- Disk
- Display Adapter
- File Packages (for SW)
- File System
- Network Adapter
- Printer
- Printer Spool Queue
- Processes
- Processor
- System

- Tape

Each component has attributes which makes up the DSIS information. DSIS compliance is defined on a per component basis. An attribute is either Basic (B) or Extended (E) compliance. To conform to DSIS Basic compliance for a particular component, the system must support all Basic attributes for that component. Likewise to conform to DSIS Extended compliance the systems must support both Basic and Extended attributes. A third category is Optional (O). Support of some or all optional parameters does not impact DSIS compliance.

The management information model for each component consists of a collection of standard and component specific attributes. The standard information set which is available to all HW components is:

Standard Hardware
- Manufacturer
- Model
- Serial and Revision Numbers
- Firmware Revision Number
- Physical Location

Standard Status
- Operational State (e.g., enabled, disabled, ...)
- Usage State (e.g., busy, idle, ...)
- Availability Status (e.g., off line, test, ...)
- Administrative Status (e.g., locked, shutting down, ...)

Standard Error
- Fatal Error Count
- Major Error Count
- Warning Error Count

Software is handled by a File Package Group. A File Package is a piece of installed software ,e.g., the operating system or data file. Standard software attributes include name, manufacturer, revision information, serial no., country code, type (e.g., O.S., application, data, other), size, install path, and license information.

EXISTING STANDARDS ORGANIZATIONS

Early on the DSIS Consortium recognized it needed to work with existing standards organizations in order to have widespread agreement and adoption of support standards. One obvious candidate was the Internet Engineering Task

Force or IETF. The IETF has developed other information standards related to networking including MIB2 for network interfaces and the RMON (Remote Monitoring) MIB for network monitors. Shortly after DSIS was formed, the IETF started a work group to define a new information standard for systems called the Host Resources MIB or HRM. The HRM working group is charted to produce a document that defines MIB objects that instrument objects common to all Internet hosts including UNIX and DOS based machines. DSIS has shared early versions of its requirements specifications with the IETF. The HRM will likely be a subset of the DSIS requirements when it becomes a standard in late 1993.

Another IETF activity of interest to DSIS is the evolution of SNMP (Simple Network Management Protocol). SNMP Version 1 (SNMPV1) has enabled the monitoring of multivendor networks by using a standard management protocol. Recently the IETF approved a more robust SNMP Version 2 (SNMPV2) which allows SNMP to be used in more robust management applications. Security has been added to SNMPV2 which is required for DSIS system management activities such as changing system configurations. Additional SNMPV2 features include support for network protocols other than TCP and large data transfers.

In the OSI world, DSIS has linked up with the Open Systems Foundation's Management Special Interest Group (OSF ManSIG) which is defining common operating system objects for OSI systems. This group will be using DSIS requirements as input to their specification.

The Desktop Management Task Force (DMTF) is an alliance developing an architecture called the DMI or Desktop Management Interface. The DMI is designed to let users easily access and manage desktop systems and all the components connected to it. It is platform and operating system independent just like DSIS, and many of its members are also DSIS members. DSIS is working with the DMTF to define the information which the DMI will use to manage desktop systems.

NEXT STEPS

At this time (6/1/93) DSIS is ready to make available phase 1 of the requirements specification for public comments. The focus of phase 1 is problem detection and analysis and includes "read only" information. Future DSIS work will include standard "write" operations to do problem resolution. Additional topics will expand the scope of information to include more components/operations to make DSIS a more complete environment for standard system management and support. With the Host Resources MIB likely to be a standard by the end of 1993, it is expected HRM conforming products will start to appear in 1994. At this time

users will start to reap the benefits of standard support information which will accelerate the adoption of DSIS information by product vendors. To get more information contact:

Raymond Edgerton, Chairman of the DSIS Consortium
Bell Atlantic Business Systems
50 East Swedesford Road
Frazer, Pa. 19355 Tel. 215 -296 6159
or
Bill Bowman 415-691 5678
Hewlett Packard,
billb@hpprsd.mayfield.hp.com

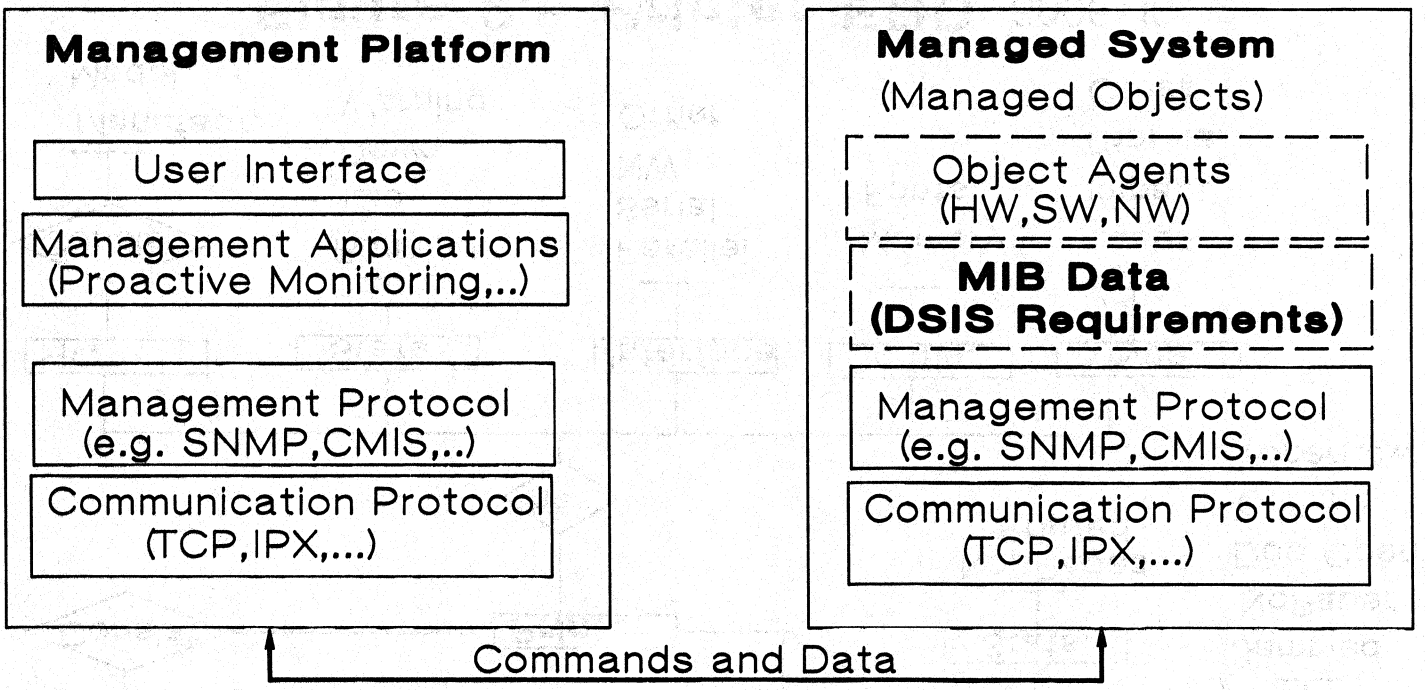


Figure 1- Management Overview 6006-9

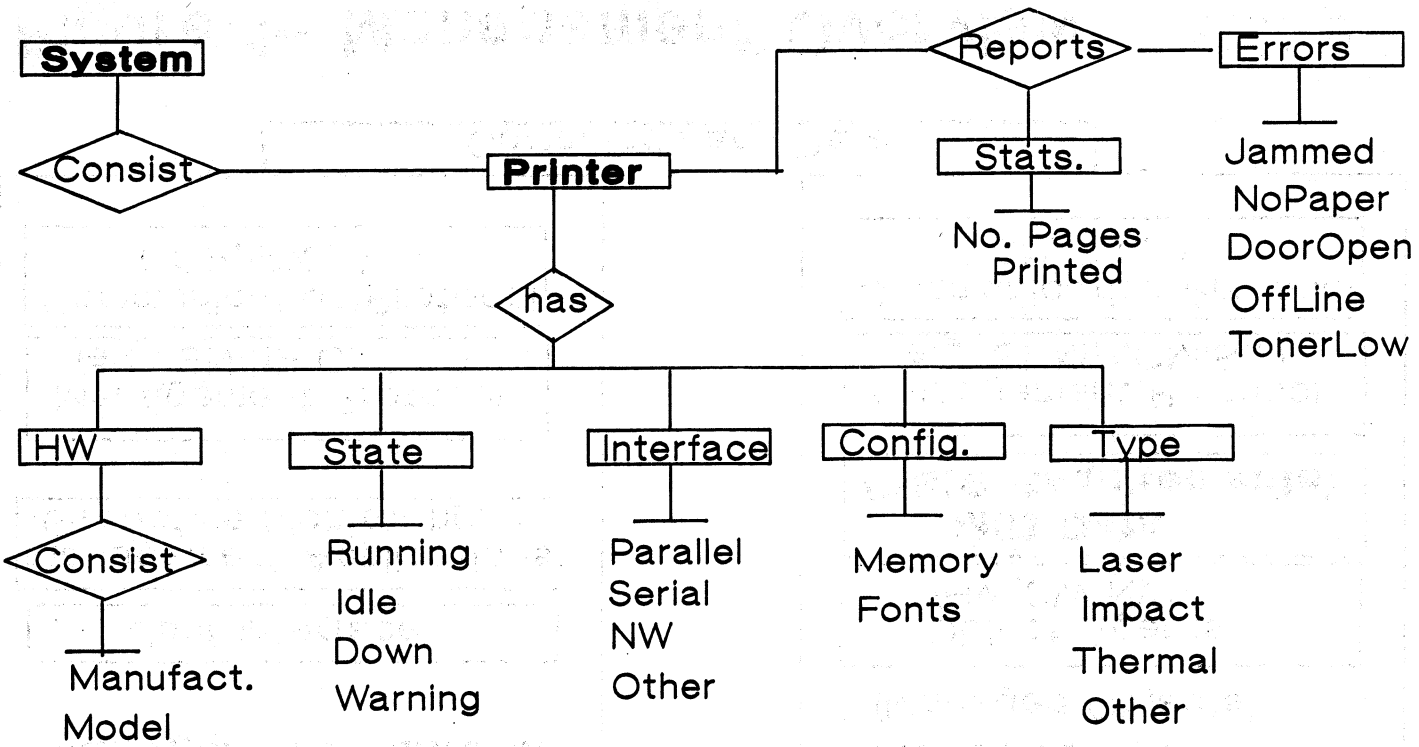


Figure 2.– Printer ERD 6006-10

Figure 3 -Printer Attributes

Item	Data Type	Repeat(R) Single (S)	Compliance
1.Printer status table (Refer to DSIS Standard Status)		S	B
2.Printer error table (Refer to DSIS Standard Error)		S	B
3.Printer hardware table (Refer to DSIS Standard Hardware)		S	B
4.CurrentDeviceErrors	Bit string: Other Jammed NoPaper DoorOpen BufferOverrun TonerLow NoToner LowPaper	S	B
5.PagesPrinted	Counter	S	E
6.NumberofSides	Enumeration: One Two	S	O
7.PrinterType	Enumeration: Other Laser Thermal Impact	S	B
8.MemorySize	Integer	S	O

Item	Data Type	Repeat(R) Single (S)	Compliance
9 .InstalledFonts	Display string	R	O
10.PrinterLanguagesSupported	Display string	R	O
11.InterfaceType	Enumeration: Other Parallel Serial Network	S	B

Compliance:
B= Basic
E=Extended
O= Optional

Using ALLBASE
for
High End, Mainframe Class Production Applications

Vish Krishnan

Computer Systems Division
Hewlett Packard
19111 Pruneridge Drive
Cupertino, CA 95014

Paper #6007

1.0 INTRODUCTION

1.1 Characterizing the High End Environment

The High End Information Management marketplace is understood and described differently by different vendors, customers and people in the academic world. Instead of presenting a critique on those definitions, the author has attempted to glean some salient points and present them here. Strictly from the standpoint of database management systems, any production environment that has one or more of the following set of requirements can be classified as a High End Environment:

. Databases tend to be extremely large. Different users have different notions of a baseline size where databases start to be "large". Generally speaking, large databases run into tens and hundreds of gigabytes. There is really no sanctity to that number. Databases smaller than that size may very well have similar requirements.

. High transaction throughput requirements are typical.

. Production environments often need to run non-stop.

. Unattended (therefore automated) system administration is an important requirement.

. Business data is often naturally decentralized. Thus, there is a requirement for some form of distributed processing.

. Not infrequently, such environments tend to drive a high number of concurrent users. As a result, database concurrency becomes a key performance factor for OLTP applications.

. And finally, some high end environments simply cannot get away from complex decision support queries (large JOINs, SORTs, aggregation oriented queries are examples of large decision support operations).

If your environment is characterized by one or more of the above requirements, please read on. You will find the discussion on ALLBASE High End features relevant to your needs.

1.2 ALLBASE and the High End Market

This paper is focussed on describing ALLBASE features that address the above requirements directly or indirectly. The following framework will be used to describe ALLBASE features:

. High End Performance Requirements
. High End Database Availability
. High End Distributed Processing
. High End System Administration

1.3 ALLBASE - a Quick Introduction

1.3.1 Overview

The information presented in this brief subsection is primarily for the reader who is unfamiliar with ALLBASE. Hewlett Packard's ALLBASE/SQL product is a relational DBMS engine for the MPE-iX and HP-UX platforms. From the standpoint of the SQL standard, ALLBASE is conformant with **ANSI SQL89 and X/OPEN XPG3.** From the standpoint of connectivity and distributed processing, it is also conformant with the Microsoft ODBC specification (an API that is currently a proper subset of the SQLACCESS standard) and with the XOPEN XA standard for communication between a conformant Transaction Processing Monitor and an ALLBASE-based resource manager. The ALLBASE feature set is extensive, and will not be listed in this paper. The reader is encouraged to get more information from their HP representative.

1.3.2 History

Over the last few years, the ALLBASE team has focussed on the following areas:

1. SQL Functionality
2. Engine performance
3. High Availability
4. Connectivity
5. Client Server Processing
6. Distributed Processing
7. DBA and System Administration/Monitoring Tools
8. High End Considerations

The earlier development years focussed on SQL functionality and performance. High Availability and connectivity followed, and we soon found ourselves in the area of standards-based distributed computing. We have also had key considerations that have spanned the entire life of the ALLBASE product so far. Notable are areas of performance and Database Administration Tools.

Starting with the F.0 release, the ALLBASE team has focussed strongly on the entire High End business, and the special database features required to support that business. It is this part of ALLBASE that this paper will present at some level of detail. The next 4 chapters are dedicated to the 4 High End items specified earlier: Performance, High Availability, Disitributed Processing and System Administration.

2.0 HIGH END PERFORMANCE FEATURES IN ALLBASE

2.1 I/O Performance

The key aspects of I/O examined here are, of course, data I/O and log I/O. We will look at the underlying philosophy of data and log I/O, and to keep it relevant to ALLBASE, we shall mix it up with descriptions of specific features.

A quick summary of data I/O is in order before we jump into any more detail. In general, data pages seldom go to disk. The idea is to minimize data I/O to the extent possible. Dirty data/index pages get posted to their respective files under the following conditions: (a) A CHECKPOINT is issued - this could be implicit within ALLBASE or an explicit action undertaken by the DBA (b) there is buffer pool pressure - in that case, candidate pages need room in the buffer pool (c) NOLOG pages need to go to disk at COMMIT time - this is a logging optimization, and will be discussed during the presentation.

Log records on the other hand go to disk under the following conditions (a) a transaction commits (b) the log buffer fills up (c) a data page is to be flushed to disk - in this case, the corresponding log records will ALWAYS go to disk BEFORE the data page goes to disk.

The ALLBASE log manager uses a ***delta logging*** approach, i.e. it logs absolutely what is necessary and sufficient - no more, no less. Further, it performs ***group COMMITs*** at the lowest level thereby minimizing the total number of I/O posts either to the underlying file system and/or to raw partitions.

From the standpoint of data I/O, the following areas have been (and continue to be) examined:

. Performance of initial table LOADs
. Performance of table/database reorganization
. Performance of table access (especially serial scans and SORT operations on large tables)
. Space Management
. Memory Management (shared memory and process local heap space)

Initial table LOAD: As the reader may be aware, an ALLBASE table (or index) may be spread across multiple (DBE)Files within an ALLBASE (DBE)Fileset. For large tables, it is necessary to achieve as much parallelism within the engine as possible. In the ALLBASE G.0 release, when multiple users issue concurrent LOAD scripts on a large table, they can instruct the ALLBASE engine to implicitly pick different DBEFiles for each user process. This ***parallel fill*** algorithm speeds up initial LOADs considerably. The results of our G.0 LOAD benchmark will be discussed during the presentation.

There is one more major enhancement. During initial LOADs (of large tables), there will be a very large number of NOLOG pages generated by individual LOAD transactions. These pages need to go to disk at COMMIT time. The G.0 release has been enhanced to achieve ***parallel posting of NOLOG pages*** at COMMIT time. Please note that since parallel posting is based on the underlying system functionality, its impact varies between our supported platforms. These differences will be discussed during the presentation.

We continue to examine the potential for greater degrees of parallelism within the product. There is a considerable amount of investigation in the works.

Database Reorganization: In general, database reorganization becomes necessary for performance reasons. There are 3 straightforward reasons for reorganizing data, and then there are other more advanced ones. The straightforward ones have to do with (a) high clustercount of B-Tree indexes (b) high overflow page count of hash tables (c) large number of indirect tuples - these are caused mainly by updates to variable length columns and on account of null-value related activity. The more advanced reasons may have to do with the need to move tables across DBEFilesets, the need to split large tables into smaller horizontal partitions and so forth. In every one of the cases listed here, table reorganization can be described as the act of *performing an exhaustive DELETE operation on a table (and therefore on associated indexes) followed by large INSERTs to an empty table or a relatively unpopulated table.* The performance of large INSERTs has been discussed under Initial Table Load. The performance of large DELETEs has also been greatly enhanced with ALLBASE G.0. We support the new ***truncate table*** feature which will delete all rows from a table with minimal logging, and will preserve all associated object definitions. Thus, TRUNCATE TABLE is somewhere between a DROP TABLE and an exhaustive DELETE from that table. The minimal logging done for TRUNCATE TABLE together with the fact that data I/O is almost completely eliminated, have resulted in substantial performance gains. Many of the readers of this paper will immediately recognize this scenario. They have all been through the pain of large DELETEs or the pain of having to DROP tables in order to speed up database reorganization.

Accessing Large Tables: ALLBASE supports multiple access methods. Further, while data access is by default an optimizer decision, ALLBASE G.0 also allows the user to tell the query optimizer which access method to use for tables in an SQL statement. The access methods available are: sequential, B-Tree index access, hashed access (based on a defined hash key) and direct access (based on row-ids).

All this is by way of introduction to data access. One important focus of our G.0 activity has been to achieve ***parallelism in sequential scans.*** Our scan algorithms have been modified to fully exploit the underlying file system page prefetch function. Again, we will discuss the results of our serial scan benchmarks during the presentation. This enhancement was benchmarked against a ported mainframe-class database environment. The comparative results will be discussed during the presentation.

Another aspect of month-end batch processing as well as decision-support oriented environments is the ***performance of large SORT operations.***. The ALLBASE SORT algorithm was completely redesigned in our E.1 (1990) release. The focus of the G.0 release was to allow user-controlled placement of SORT-related spaces. This will be discussed under the heading of space management.

Space Management: For high end environments, data, index and log space needs to be totally manageable by the user, DBA, system administrator and so forth. The following considerations are critical:

. Total control over placement of files across devices
. Total control over placement of tables and indexes
. Efficient algorithms for space search and allocation during INSERT operations
. Support for transient space allocation (typically for SORT operations)

The ALLBASE space architecture was fundamentally designed to be flexible in terms of the placement of data and index files, the separation of data from index, the separation of data/index from log file(s), the placement of tables and indexes in specific filesets and files and other associated activities where control over placement and allocation is desirable.

There is also the issue of space search and allocation every time a new row is inserted into a table/index. On High End systems, where DBEnvironment sizes may run into gigabytes, page/search allocation schemes are all the more performance critical. The ALLBASE storage manager has historically had efficient algorithms in that area. In the G.0 release, as part of our ***High End Space Management***, we have greatly speeded up ***space search and allocation*** for table and index INSERTs.

In the 1992 release (F.0), ALLBASE was enhanced to provide better performing SORT-space management. Some background information is essential here. Inside the ALLBASE engine, the following user actions may prompt a SORT:

. SELECT .. ORDER BY
. SELECT .. GROUP BY
. SELECT DISTINCT
. SELECT .. UNION
. SORT MERGE JOINs

In all the above cases, ALLBASE will avoid a SORT altogether if appropriate indexes are available for relevant scans. If a SORT is necessary, two kinds of transient space may be required:

. Temporary Space for sorting intermediate runs
. Temporary Space to contain the resultant set of rows to be sorted (in place)

Space for intermediate runs can be allocated through the ***create tempspace*** command. Tempspaces thus created can be directed to any device available to the environment. Further, temporary space for result tables will default to the SYSTEM DBEFileset unless redirected through a ***user-controlled sort-fileset*** specification. This features allows concurrent SORT operations to run without interfering with each other's space allocation.

Shared Memory and Heap Management: The amount of global and process local memory available to database processes has a direct impact on I/O performance. In ALLBASE, shared memory consists of the data buffer pool, log buffer pool and control block space. Heap memory is used by process local activities e.g. memory for SORT operations, memory for cached queries and stored procedures etc. The ALLBASE engine uses one basic principle for Shared Memory Allocation: the size of memory available should be limited by the system as a whole, and not by the DBMS engine. Thus, within the bounds of a short-pointer address space, the ALLBASE engine now supports ***shared memory sizes limited only by system considerations.***

In terms of heap management, ALLBASE uses ***process-local mapped objects*** to the extent possible. In the MPE-iX context, this translates to SR5 space, while on HP-UX, we use the UX 8.0 enhancement for memory-mapped short-pointer space whenever we can.

There is one more consideration. With such large shared memory sizes available, optimized ***buffer-search algorithms*** are critical. The ALLBASE storage manager uses efficient structures and search algorithms for buffer pool pages as well as run-time control blocks.

In summary, the ALLBASE engine is fundamentally designed for industry-strength I/O performance. Our 1993 release (G.0) provides even stronger features for the specific areas discussed above.

2.2 Multiprocessor Scaling

Hewlett Packard is known industrywide for its scalable PA-RISC mulitprocessor architecture. In this discussion, I will briefly describe the main issue with DBMS scaling.

All DBMS engines have some fundamental internal or user-controlled operations that may become points of serialization. Such events naturally detract from the ideal scaling factor achievable. It is therefore critical that such events be made as efficient (i.e. reduced path length) as possible. Further, DBMS systems need to be able to break up some of these events into smaller sub-events that execute in parallel.

The 2 aspects of scaling that apply to all DBMS engines are ***latches*** and ***checkpoints.*** A latch is an in-memory synchronization primitive (it can be thought of as a short duration lock on shared memory structures - that is really what it is). Checkpoints are required to post ALL data pages and corresponding log records to disk.

It is not inconceivable that data/index locks become points of serialization. Even a large environment with random OLTP accesses may have hot spots. The author would like to emphasize here that where data and index related hot spots are concerned, ALLBASE has enough methods to minimize (if not eliminate) such points of contention through physical/logical design, transaction architectures and system load balancing where appropriate. We will be dealing with these issues under the subheading of concurrency control.

The ALLBASE latch manager has forever been an extremely lightweight algorithm. The fundamental ingredient of all synchronization in ALLBASE is the compare-and-swap primitive which is implemented through the LOAD-AND-CLEAR-WORD instruction in PA-RISC. Further, the ALLBASE latch manager uses an efficient hash algorithm for the necessary in-memory searches as well as for eliminating all wait-states other than the minimally necessary ones.

A lot of attention has been paid to latch-wait states across the board. The details of it all will not be described in this paper. However, certain examples stand out. One of them is the optimization around the log-force event. At the lowest levels in the storage manager, log-force activity has built-in latch optimization to ***minimize latch serialization and process dispatches.***

CHECKPOINTs in ALLBASE are implicit as well as user-driven. An implicit CHECKPOINT happens under one of several conditions. A LOG FULL triggers a CHECKPOINT and so do the internal BEGIN/COMMIT ARCHIVE functions during an online backup. At any rate, a CHECKPOINT may cause several data/index pages to go to disk. This is much more the case on the High End where large buffer pool sizes are common. The ALLBASE engine employs a ***parallel post*** algorithm across multiple DBEFiles that may be involved during the CHECKPOINT. It should be noted here that parallel posting is dependent on underlying features available on the system, and as such, it works differently on different platforms.

2.3 Concurrency Control

This subsection deals with the issue of locking. We will approach it from two different angles. Firstly, locks have 3 simple properties: the kind of lock (shared or exclusive), the granularity of the lock (table, page, row level) and the duration for which the lock is held once acquired. Secondly, the whole issue of locking is closely tied to the access method(s) employed by the query optimizer or by the user. If serial scans are used, tables get locked in a way probably different from if B-Tree index access is employed.

Additionally, deadlock minimization (hopefully elimination) is a constant goal for production environments. And then, there is the entire issue of system monitoring and control of locking

activity. This is especially critical since ALL production shops will encounter the occasional pathological SQL activity that will tend to run away with system resources, locks in particular.

In terms of lock properties, the ALLBASE storage manager will use a kind of lock that is necessary and sufficient for a given user action. Under some conditions, it will even perform READ operations without acquiring any locks whatsoever. In addition, ALLBASE allows the user to control the nature of table locks acquired through the LOCK TABLE command.

As for locking granularity, users have full control over table, page, row locks. More importantly, these decisions can be changed dynamically for user tables and/or system catalog tables.

Proper use of isolation levels is critical to High End systems where high concurrency is a requirement. Even for environments with not such high throughput requirements, specific situations may demand optimal use of isolation levels. As an example, some OLTP environments have the occasional large decision support query that needs to execute concurrently with minimal impact on OLTP throughput. ALLBASE provides an exhaustive set of transaction isolation level options that can be controlled at the user level. In the G.0 release, the product has been enhanced with a ***set isolation level*** feature that allows for session-global as well as transaction-local settings of transaction attributes. This feature further enhances user control over isolation level selection.

Concurrency control is very closely related to the access methods selected for participant tables in a query. For instance, if a table is being accessed sequentially, the storage manager is clearly going to touch all rows. It will therefore acquire a table level lock and no other locks. If however, an index scan is deemed optimal, the storage manager will only lock the relevant pages/rows from the table. The question is: what kinds of locks will ALLBASE acquire on index pages/rows? In the F.0 release (1992), we modified the ALLBASE engine with our ***concurrent B-Tree*** enhancement. Stated simply, ***index readers read without locking index pages/rows at all, and index writers write without acquiring exclusive locks on index pages/rows.***

This approach represents a fundamental shift from the way we have dealt with index concurrency in the past. In the High End OLTP environment, index access is common. Tuple INSERTs, DELETE and key column UPDATEs all contribute to index write activity. It is therefore extremely critical to ensure that indexes do not become concurrency hotspots. The concurrent B-Tree enhancement achieves that.

On the subject of deadlocks, the ALLBASE policy is to minimize it if not eliminate it. Complete elimination of deadlocks is really dependent on the way the production system is or can be designed for that purpose. In terms of engine-specific features, ***intention locking*** in ALLBASE already provides a baseline for minimal deadlocks.

One major focus of the ALLBASE group is to assist user environments in quick and clear detection of deadlock participants. The High End application environment will typically have hundreds of concurrent users running volumes of SQL code with numerous SQL queries. ALLBASE users may enable the ***deadlock matrix display*** which will dump out details of deadlock participants to a user-specified file. The performance monitor for ALLBASE can then be used to analyze this dump.

Yet another area that mainframe-class applications are very conscious of is the entire issue of resource control. With the best of controls, there will be some rogue transaction or query commandeering shared memory through excessive locking. The main goal there may seem to the early detection of such outlaw activities. There is however, no substitute for prevention of such events. The ALLBASE engine allows DBAs to set ***lock control limits*** for individual server processes as well as for the database environment as a whole. These specifications will limit sessionwide and systemwide consumption of lock control blocks. Mainframe-class environments find this feature quite indispensable.

On the subject of providing system level controls, ALLBASE F.0 introduced the ***transaction level timeout feature.*** The Database Administrator may set a database-global timeout value. In addition to (or instead of) that, timeout values may be dynamically set through the F.0 ***set timeout*** command.

2.4 Synchronization

The issue of synchronization for database server processes can be defined in terms of 2 key elements: primitives for synchronization (latches and locks) and waiting/waking up. The essential synchronization primitive is the compare-and-swap operation. This is achieved through the PA-RISC instruction LOAD-AND-CLEAR-WORD.

Very briefly, ALLBASE employs latches for updating and accessing in-memory elements like linked-lists and certain specific counters. As for locking, when a lock request is generated, the lock manager needs to quickly check if the lock already exists. Lock searches are bound to be frequent, and therefore need to be supremely efficient.

Once a lock/latch request fails (because some other user has locked/latched that object), the requesting process needs to wait. Further, waiting processes need to be woken up once the wait-state is resolved.

It has been stated before that the ALLBASE latch manager is designed for high end performance and scaling. Additionally, ALLBASE makes use of efficient hash-based lock search/allocation. In our 1992 release, we have further tuned our wait/post mechanism. The highlight of this enhancement has been ***optimized posting of wait-queues.*** This is a new algorithm that adopts a lightweight single-call scheme for its posting mechanism.

It goes without saying that ***semaphores*** are the basis for wait/post activity on HP-UX systems. It should also be pointed out here that process synchronization, like some I/O related enhancements, is platform-specific.

2.5 Access Methods and the Query Optimizer

As stated earlier, ALLBASE supports 4 access methods: sequential, B-Tree index, hashed access and direct tuple access. From the standpoint of High End data access, the following activities are noteworthy.

The ALLBASE optimizer is cost-based and statistics-oriented. Even with the best statistics, optimizers sometimes honestly do the suboptimal thing. Such anomalies are especially critical in High End environments where the impact of choosing a suboptimal JOIN order (just as an example) will be much higher. The ALLBASE G.0 release will provide the ***access plan modification*** feature that will allow users to override optimizer decisions.

From the standpoint of index access, ALLBASE already provides ***multikey access*** as part of its OR optimization. Consider the following SELECT statement:

```
SELECT ..... FROM T1 WHERE
             C1 BETWEEN 10 and 20
             OR
             C2 BETWEEN :hv1 and :hv2
```

Assuming that C1 and C2 both have cost-effective indexes on them, the ALLBASE optimizer will perform 2 distinct index scans and then do a no-duplicate UNION of the two sets. This is the essence of multi-key access. The ALLBASE team is working on further optimizing this feature.

Before we leave the subject of optimizer functionality, a brief discussion on High End decision support activity is worthwhile. It is not at all uncommon to see large ad hoc SELECT statements in such environments. The relevant attributes of such queries are (a) they tend to have a large number of participant tables (b) they tend to be dynamic, i.e. quite possibly, every invocation of such queries is causing them to be re-compiled (and therefore re-optimized).

Such large, dynamic queries can be speeded up through the use of ***semipermanent sections*** (this allows for repeatedly used dynamic queries to be cached) , the use of ***dynamic parameters*** (this allows the same query template to execute with varying parameter values WITHOUT using up extra process heap space), and a few other miscellaneous ALLBASE features.

In ALLBASE G.0, the optimizer has been tuned to further speed up optimal plan generation through ***enhanced pruning of candidate access plans.*** Typically, for a large JOIN, the number of plans to be evaluated by the optimizer may grow rapidly as the number of tables in the JOIN increases. This is especially critical for large dynamic environments, and for static SQL environments currently under development. Users in such environments will see faster query preprocessing (especially for large JOIN queries) with ALLBASE G.0.

2.6 Client-Server Features

The ALLBASE query processor has historically supported both the static and dynamic query paradigms. In that model, the unit of communication between the client layer and the database layer is a single SQL statement. For remote clients, it is really critical to minimize interprocess communication over the network. The day is not far when OLTP applications will be expected to provide mainframe-class OLTP throughput between non-terminal based clients and database servers. (we are probably already there). ALLBASE is extremely well suited for environments of that nature. The following features are notable:

Client applications can make use of ***stored procedures*** in ALLBASE. A stored procedure is a collection of SQL statements with support for conditional syntax and semantics. When a stored procedure is thus created, it is compiled into the database itself. At run time, the client application need only invoke the stored procedure (with appropriate parameters). This results in multiple SQL statements getting executed in a single server invocation. Stored Procedures also have a serious productivity benefit in that they allow for centralized encapsulation. As changes are required to procedure logic, they need only be made to one place.

Another feature to make note of is ***business rules (also known as triggers or assertions).*** Users may defined rules into the database so that the engine may automatically take a specified set of actions upon the occurrence of certain INSERT/DELETE/UPDATE events. In the absence of rules, client applications would have to implement the same actions in user code. This is a performance issue (more calls to the server) as well as a productivity factor.

An emerging area of client server processing is that of ***multithreaded application server environments managed through Transaction Processing Monitors.*** A typical application server environment will consist of a remote client that is primarily involved in presentation management,and does no direct SQL or database manipulation. Such a client interacts with the remote a database application+server through ***function shipping***. (Functions of this nature will invoke a set of application procedures or stored procedures on the server system). The number of instances of various applications on the server system is controlled by TP Monitor-like products. Application server instances could be disjointed processes (PINs or PIDs) or they could be ***application threads.*** Depending on the way threads are implemented on the platform,

they may have the effect of reduced process management work on the server system. In general however, the main impact of application threads is ***enhanced concurrent processing of a large number of client requests.***

Database server engines need to ensure thread-safety in order to benefit from the performance of multithreaded applications. There are memory implications as well as session management issues. ***Thread-awareness is part of the ALLBASE G.0 functionality.*** Again, the incorporation of thread-safety is platform-specific. This will be discussed during the presentation.

2.7 Summary of High End Performance Features

The ALLBASE group has been involved in extensive enhancements in the performance arena especially where it concerns High End environments. The reader is encouraged to identify the area(s) specific to his/her business and production shop(s). Please feel free to contact your HP representative for more information.

3.0 HIGH END AVAILABILITY FEATURES IN ALLBASE

3.1 Introduction

There is a commonly accepted definition of High Availability, but different sources colour it just a bit differently. For a common understanding between the writer and the various readers, let us just define it here. High Availability here deals with allowing ***non-stop operations through the elimination of planned down time.*** It gets better. High End environments place some very specific and rigorous demands on High Availability. As we shall see in the following subsections, the existence of super-large databases, a large number of concurrent sessions, high transaction throughputs, and long periods of non-stop operations all add up to a sizable High End set of requirements for High Availability.

3.2 High Availability for all - features for basic planned outage

Let us examine some common events that may require planned outage.

- Data backups
- Log backups
- Space Management (log and data)
- Mirrored log files

ALLBASE supports ***full online backup of data and log files*** on both MPE-iX and HP-UX platforms. It should be noted also that ALLBASE on MPE-iX makes use of TURBOSTORE-II underneath, and further, such an online backup session DOES NOT require any initial down time.

Almost more important than online backup of data is the management of log space. In an archival non-stop environment, applications MUST be able to continue through archival LOG FULL conditions. ALLBASE supports ***implicit SWITCH LOG*** whereby users may specify multiple log files in the environment. When the current log fills up, the ALLBASE storage manager will automatically switch over to the next log and so forth. The main purpose of the

SWITCH LOG feature is NOT just to allow more log space. It is intended to free up a log file so it may be backed up. The whole idea is to allow for an ***unattended backup scheme***. The Database Administrator may specify up to 34 log files in the environment, but for non-stop archival operations, a minimum of 2 log files is needed. As log L1 fills up, ALLBASE will switch to L2 thus freeing up L1 for backups. As log L2 fills up, ALLBASE will attempt to switch back to L1. This will be allowed if and only if L1 has already been backed up. So, it is critical that user environments set up background backup processes to be looking out for logs ready to be backed up. ALLBASE enables such activity through the ***STORELOG (online) command in SQLUTIL.***

Even with the log backup scheme in place, any production environment based on the above philosophy is still dependent on human presence for operator activities such as tape mounts. The HP system environment supports large capacity backup devices such as ***magneto-optical devices*** that provide enhanced unattended operations. Also of interest are ***pseudo-operatorless environments*** where there will be an operator once a day or once per shift. In such cases, it becomes quite necessary to ensure that a log file IS ready to be backed up when there is a human being present to mount a tape. The ALLBASE product also supports ***user-controlled SWITCH LOG (CHANGELOG)***. This feature may be used effectively to ensure that log files are ready to be backed up when the operator is present.

The above discussion gives the reader a brief idea of issues related to data and log backups and the features available in ALLBASE for the same. Let us now move on to the subject of space management. The ALLBASE environment has the concept of a DBEFileset that consists of physical (DBE)Files. A table or index belongs to a DBEFileset. When an INSERT or UPDATE to a table fails because of a "no more space" exception, the DBA merely needs to create another DBEFile (of the correct type) and add it to the appropriate DBEFileset. Thus, space management in the ALLBASE product is fundamentally dynamic.

Be that as it may, the business of running out of space in the middle of a mission-critical application does not sound very wholesome. Nor is the idea of preallocating large chunks of space very appealing. ALLBASE supports ***dynamic space expansion*** which allows designers to create DBEFiles with an initial size, a maximum size, and an increment size. Files may grow from 2 pages to 512K pages (a page is 4K bytes). In the presence of multiple dynamically expandable files in a DBEFileset, ALLBASE will employ an equitable expansion scheme. In summary, dynamic space expansion lets production environments run smoothly without encountering a "no more data/index space" condition, and, equally importantly, without pre-allocating large amounts of initial DBEFile space.

On to the subject of ***log mirroring.*** In the database world, mirroring of data and log are employed to provide an element of fault-tolerance. We have not defined this term previously. The discussion on ***fault tolerance vs High Availability*** can be taken up during the presentation. Simply stated, fault tolerance is ***the minimization or total elimination of unplanned outage (down time).*** The author of this paper is taking the liberty of introducing fault-tolerance features under the subject of High Availability.

From the standpoint of total recoverability, the database log file is far more critical than the data files. In case of media failure, the installation can go back to any old data backup, and so long as all intervening log files are available (between the point the data backup was taken and the current point in time), recovery will happen. Recovery time will of course be dictated by the vintage of the data backup. The point is that if some intervening log file is missing, recovery will not be possible beyond that point. So, it is absolutely critical that environments that care for such eventualities put enough redundancy around log files. ALLBASE provides a ***DUAL LOG*** feature that allows users to have 2 identical logs on (hopefully) two different partitions/volumesets. For MPE-iX users interested in disk mirroring of logs only, ALLBASE also allows users to create ***the log file in a different volumeset*** from the data DBEFiles.

Thus, log mirroring can be achieved through either the ALLBASE DUAL LOG feature or through the system-level disk mirroring products (on both MPE-iX) and HP-UX systems. Further, MPE-iX users may restrict the mirroring feature to the log ONLY by placing the log on its own volumeset and then mirroring only that volumeset.

3.3 High Availability Features specific to the High End

The features described in the previous subsection constitute the basic ALLBASE High Availability offering. From the standpoint of high end environments, many other factors come into play.

Database sizes tend to be so large that full data backups (even online backups) are impractical.

Backup speeds become critical, and parallel backups, if possible, need to be exploited.

Large numbers of concurrent transactions (especially OLTP) and sessions need to be supported.

On account of non-stop operations, certain internal counters may overflow and cause the system to halt. Such events need to be minimized to the point that they become non-issues.

There are environments where applications need to go on even in the event of media failure with total data loss. Alternative solutions (like replicated sites) provide a very viable and attractive solution for constant application availability.

ALLBASE addresses the above needs. Some of the features discussed below are available with ALLBASE G.0, and the others have been around for some time. The following discussion presents the relevant feature set.

ALLBASE G.0 supports ***partial STORE, partial RESTORE and partial rollforward***. The recommendation is to identify high traffic components of the database, and perform backups on only that subset. The SQLUTIL STORE, STOREONLINE and RESTORE commands have been enhanced to provide ***STORE/STOREONLINE (partial) and RESTORE (partial).*** The whole idea behind partial STORE and RESTORE is that in the event of media failure, the appropriate backup is restored and rollforward recovery needs to be performed on ONLY that component of the database.

The SQLUTIL SETUPRECOVERY command has been enhanced to support ***SetupRecovery (partial).*** This command essentially allows for a subset of the environment to be recovered ***while users continue to access the rest of the database environment.***

This begs the question: how does the user really know which DBEFiles/DBEFilesets are likely to fail, i.e. what components of the database environment should be partially STOREd? The way the entire partial operation has been designed and externalized, the following approach will work in all cases:

. At some point in time, the entire database environment is backed up.

. Following that, certain DBEFiles/Filesets are identified as high-traffic components. These are then backed up on a regular basis through the ***store (partial)*** command.

. In the event of a corruption, ALLBASE will report back the identity of the DBEFile that needs to have rollforward recovery performed on it.

. Once the DBEFile is known, the user would be well advised to ***restore (partial)*** the most recent backup of this DBEFile. If no backup is available since the full backup, that will also work. The way RESTORE PARTIAL works, it will extract the necessary components even from a full backup.

. The restored DBEFile may now be rolled forward using the ***setuprecovery (partial)*** scheme.

. The repaired DBEFile may now be reintroduced into a running environment through the use of the new ***DETACH and ATTACH commands in SQLUTIL.*** These commands allow for DBEFiles and DBEFilesets to be "removed " from and "added" into running environments without interfering with any database activity currently in progress.

ALLBASE G.0 also allows users to explicitly render certain parts of the database inaccessible through the **DETACH** feature in SQLUTIL. This feature is primarily useful as an enabler to partial rollforward, i.e. before a piece of data can be recovered, it must be first made offline. However, the command by itself is extremely useful outside of the rollforward situation. Consider a large multimedia database where documents are stored on jukebox platters. These documents are very infrequently accessed, and once stored, are never updated. It is conceivable that the system administrator wants to remove the data from the scope of day-to-day administration (e.g. full online backup). This will also cause total steady-state storage requirements to drop significantly.

With ALLBASE F.0, we have also started to exploit backup features available through system backup utilities. For instance, TURBOSTORE-II allows the user to specify ***a set of backup devices to be used in serial or parallel or any user-specified combination thereof.*** As part of SQLUTIL's STORE, RESTORE or STOREONLINE commands, ALLBASE users can specify the same kind of syntax for parallel backups or serial backups. Note that the parallel specification enhances the availability of the entire system through faster STORE/RESTORE . On the other hand, a serial specification of devices enhances the operatorless nature of the environment.

The ALLBASE/HP-UX user has the option of using the HP-UX OMNIBACK utility to store/restore ALLBASE DBEnvironments and log files.

Moving right on to support for a ***large number of concurrent transactions and sessions.*** The engine does not set any limits on the maximum number of concurrent sessions on a DBEnvironment. The only limits that ever existed were in the user's ability to display the attributes of connected sessions. These limits have been eliminated through ***enhanced pseudo-table access*** - a feature in ALLBASE G.0. As for supporting a large number of concurrent transactions, user environments no longer need to code for the "Max transactions exceeded" error. With the ALLBASE F.0 ***transaction throttle*** mechanism, the transaction manager will internally enforce the maxtransaction upper bound without requirng user applications to worry about that exception condition.

In the High End marketplace, it is not uncommon for a production environment to run for long periods of time without ever taking a break. This has the potential of encountering counter overflows. In general, transaction uniqueness (and therefore recognition of correct page states) is governed by versioning schemes that need to minimally depend on some monotonically increasing field somewhere. With ALLBASE F.0, we have totally enhanced the ***scheme for generating unique ids for transactions and data/index/log versions.*** A quick calculation done on the new scheme showed that in one of the worse cases, it will be over 50 years before

the environment will see a version/transaction id overflow. Needless to say, the whole thing gets unique again upon the next warmstart of the database.

The non-stop argument applies for statistical counters as well. Such counters, although they are not nearly as critical as their page version counterparts, can be painful when overflows happen. With ALLBASE F.0, we have also revised our ***table/index statistics computation***. The key stats in question here are number of rows and number of pages in a table or index.

With growing database sizes and increasing transaction rates, one might pose the question: what if the total maximum allowed data and log space is exceeded? In other words, what is the maximum data/log size supported by ALLBASE?

The answer can obtained from the APPENDIX in the ALLBASE Database Admin Guide. But the key limits are stated here. An ALLBASE DBEnvironment can have a ***maximum of 32,767 DBEFiles***. Each DBEFile can have a ***maximum file size of 2 gigabytes.*** This gives us a ***maximum DBEnvironment size of 64 terabytes.*** At this point, the ALLBASE team is not worried about customers hitting this limit.

As for log sizes, an ALLBASE DBEnvironment may configure in a ***maximum of 34 log files*** in its log directory. Each log file may have a ***maximum file size of 2 gigabytes.*** For the ALLBASE G.0 release on MPE-iX, we might increase this limit to 4 gigabytes. Independent of that, the current ALLBASE environment will support a ***maximum log space of 68 GBytes.***

And finally, let us discuss the various issues related to total or partial data loss in an environment. In the ideal case, application environments currently making use of a piece of data should be allowed to continue even in the situation of the data becoming unavailable. Consider the case of a pre-G.0 ALLBASE database where all the DBEFiles are scattered across volumes in the same volumeset. Even with all the support for highly available partial rollforward, if the master volume of this volumeset became unavailable, the whole database would be rendered inaccessible. It is therefore important for the DBMS to allow DBEFiles to reside on multiple volumesets (i.e. MPE groups). The ALLBASE G.0 allows users to have ***DBEnvironment files on multiple volumesets within the same account.*** This is obviously an MPE-iX specific issue. For ALLBASE/HP-UX, DBEFiles have always been allowed to go across partitions.

The above feature allows for High Availablity in the event of master volume failures on MPE-iX. We still need to deal with the question: what happens if the total database environment becomes unavailable for some reason? Do applications have to just stop and wait for it to become available?

One possibe response to the questions above is ***System High Availability features.*** Examples of these would be ***Disk Mirroring*** and ***Processor Switchover.*** These are powerful solutions that work at the total system level. It should be noted here that databases impose their own special requirements over and above the scope of disk mirroring and switchover. The following 2 points should be noted:

. The systems across which mirroring or switchover can occur need to be within a certain physical distance of each other.

. Following a hard system failure, the mirror database needs to somehow ensure that all available committed transactions are completed on the mirror BEFORE allowing application switchover.

Requirements of the above nature seem to hint at the fact that perhaps we need something special by way database mirroring. This brings us to the subject of data replication in general, and ALLBASE/REPLICATE in particular.

3.4 Data replication - ALLBASE/REPLICATE

From the brief discussion above, it should be clear that databases have their own unique perversities in terms of mirroring and switchover. In a High End environment, systems will very likely be geographically separated from each other. Because of the mission-critical nature of high end systems, co-located mirrors do little to allay Corporate paranoia regarding human vandalism and acts of God alike. What is required is the ability to create ***shadow databases that are geographically far apart, and can stay in real-time sync with their primary counterparts.*** Thus, in the event of a primary failure, currently connected application environments can be architected to switch over to one or more shadow or slave ALLBASE database environments.

ALLBASE/REPLICATE is an HP product positioned for High End Availability as well as the High End Distributed Processing market. In this section, we will discuss the Availability function. The next chapter deals with Distributed Processing, and the reader will see more of REPLICATE in that section.

ALLBASE/REPLICATE was first released in August 1990. Since then, its impact and applicability are being recognized by a growing number of customers. Briefly, this product has 2 components:

1. The REPLICATE engine (totally integrated with the ALLBASE engine) provides the foundation for replicate-enabled primary databases. The key to replication is the fact that it is log-based, and as such, it is persistent (Durable for readers familiar with the ACID properties of transactions). The log-based nature of replication enables it to exploit the already powerful logging/recovery machinery in ALLBASE.

2. The REPLICATE application provides the mechanism for:

   . Allowing shadow databases to come up to date with the primary WITHOUT being required to run all the time.

   . Allowing shadow databases to be subsets of the primary environment(s).

   . Allowing primary and shadow databases to get an audit trail of SQL activity.

The above are a few important functions of the REPLICATE application.

ALLBASE/REPLICATE also provides benefits in the arena of High End Performance, System administration and Distributed Processing. Very briefly, the performance impact can be seen through application load balancing. Clearly, an OLTP environment can offload its read-only activity to one or more shadow databases, thus freeing up the primary system for critical transactions. Also, if a large environment is such that applications primarily run against well-defined subsets of data, the environment can be split into smaller databases that can reside on different machines. This can then be augmented by having one large central site that is the sum-total of the smaller databases, and derives its updates through the replication mechanism from the smaller (primary) environments.

From the standpoint of High End System Administration, we now have examples of customer environments that use REPLICATE as a remote backup mechanism. That is what REPLICATE really is - a remote backup that runs continuously, or every now and then. (It is all up to the user environment and how it wants replication to be set up). In all these cases, the use of REPLICATE as a backip vehicle has a significant impact on the required ***unattended system administration*** for the High End shop.

And finally, ALLBASE/REPLICATE has become the preferred distributed processing solution for some ALLBASE customers. We find from our own experience (and industry watchers confirm it to us) that replication is usually a friendlier solution where distributed transactions are desired. It is NOT a substitute for 2-phase COMMITs. Indeed, it cannot be that. The replication model is one of asynchronous transmission in the background, and in part, its value proposition is that the secondary environment does NOT have to be up and running along with the primary environment. Replication can be real-time, but it is never synchronous. Thus, where atomic COMMITs are required across databases, replication does not seem to be the answer. This would seem obvious given the difference between the 2-phase COMMITs and asynchronous replication.

However, the theoretical argument needs to be examined in the light of practical considerations. Several of our customers have come around to believing that their previously stated need for atomic transaction commits across databases was perhaps somewhat overstated. In many cases, this has resulted in the customers' re-evaluation of their original rigorous goal. We are now finding that some customers are attracted to replication because (a) it more or less solves their original intent (b) it wins on account of its powerful set of benefits and its relative ease of production usage.

In the following chapter, we will look at distributed processing at a high level, and following that, we will examine the applicability of ALLBASE for various paradigms of distributed processing. We will naturally focus on the components that pertain to High End environments.

4.0 DISTRIBUTED PROCESSING WITH ALLBASE

4.1 Introduction - Types of distributed processing

Distributed computing is yet another one of those terms that get variously defined and used by the vendor and user communities. The following may very well be the author's point of view, but an attempt has been made here to capture the various paradigms that popularly get talked about as distributed environments:

1. Remote database access is often described as distributed computing.

2. If a client application needs to access more than one database, it sometimes constitutes distributed processing for some.

3. Environments where a large number of clients are allowed database access through a much smaller number of server processes sometimes get classified as distributed environments. Such environments are characteristic of multithreaded database/application server architectures.

4. Environments that employ data replication in any way, shape or form are often considered to be distributed.

5. Environments that use 2-phase COMMITs (truly distributed transactions) are the very epitome of distributed processing.

6. And finally, many environments think of distributed queries (JOINs) as one of the crown jewels of distributed processing.

Of the various descriptions stated above, items 4 - 6 are most popularly quoted by serious students of distributed processing technology. These areas also pertain to High End processing, and therefore, they will be the focus of this section. The arena of replication has already been detailed in the previous chapter. In the following section, we will focus on the rest, especially the topic of truly distributed transactions.

4.2 ALLBASE Support for distributed transactions

With F.0 (1992), ALLBASE provides full ***two-phase commit support.*** Moreover, this enhancement to the product has been done in conformance with the ***X/Open XA*** standard for communication between Transaction Processing Monitors and Database engines (resource managers in XA parlance). Let us spend a few words describing what all this means.

The general implementation of distributed transactions has been achieved through ***Transaction Processing Monitor*** products. The basic functionality of TP Monitors consists of support for global transactions. This involves the ability to begin a global transaction, the ability to do 2-phase COMMITs (i.e. PREPARE followed by a COMMIT) and the overall support for a decision log. This log is crucial for exception handling in the case where participant resource managers fail during a PREPARE of a COMMIT, and then wish to unilaterally connect to their relevant databases. A rather terse description that was, but we can go into more detail during the presentation.

When a user application begins a transaction in a distributed environment, it is usually up to the TP Monitor to recognize that as a global transaction and act accordingly. Likewise, when a user application commits a distributed transaction, it is up to the TP Monitor to understand and execute the 2-phase nature of the COMMIT. Clearly, the TP Monitor needs to communicate with the underlying resource managers for the above purposes.

Consider a case where a distributed transaction goes across an ALLBASE environment and a few non-ALLBASE databases. This situation now requires the TP Monitor to communicate with resource managers of different types. In the ideal case, the TP Monitor should be able to talk to the servers in exactly the same way and have them all understand and execute those commands. The X/Open XA standard specifies that procedure interface and protocol. It thus provides the foundation for ***heterogeneous distributed transaction support.***

ALLBASE F.0 can now participate in XA-conformant distributed transactions.

But the ALLBASE engine goes beyond the XA specification. While XA does not involve itself with multi-threaded client-server issues, ALLBASE has already incorporated thread-awareness as part of its G.0 functionality. This is a key enabler for database servers to participate in multithreaded activity.

In closing, a few words about existing Transaction Processing Monitors is quite in order here. A couple of XA-conformant TP Monitors exist today. The ENCINA family of products from Transarc Corporation, and TUXEDO from USL (now Novell) have already been announced. The ALLBASE team has spent a lot of time testing with ENCINA. In the near future, we will also be

testing it with TUXEDO. In general, transaction processing monitor products support distributed transactions, multithreaded client-server execution, and, at the same time, provide enablers for parallel transaction processing. This paper will not go into those details.

4.3 Distributed queries

And finally, a few words on the support for distributed JOINs. The ALLBASE team continues to investigate this functionality. No firm plans can be communicated yet. In some specific cases, our experience has been that replicated tables act as a very good substitute for distributed JOIN support. There is the classic example of a Corporate database along with several regional databases for this large Company. There may be a need at headquarters for joining tables between 2 regions. If the regional databases happen to be all replicated into the Corporate environment (or at least if the relevant tables are so replicated), the distributed JOIN becomes a local activity. The author is by no means suggesting that replication is a practical solution for every such distributed JOIN requirement. But the fact remains that some of our customers have adopted this tactic.

5.0 HIGH END SYSTEM ADMINISTRATION

Broadly speaking, production environments deal with 2 sets of problems: application environments and system management environments. Application environments have been quite suitably addressed by the ever-growing list of solution providers. By contrast, industry-strength system administration solutions have been few and far between. Before going into the technical details of sys admin activities, a working definition is required.

For the purpose of this paper, we shall define system administration as the collection of all key monitoring activities and operator functions.

Database elements that lend themselves to monitoring are (for example):

1. Data/index space usage
2. Log space usage
3. Critical transactions and queries
4. Statistics (table/index)
5. Total database performance

Operator functions will typically involve the backup and restore of data and log files. They will also involve work related to rollforward recovery, should one become necessary. Further, periodic verification of (literally) the physical well-being of database environments is a potential operator undertaking.

As per our definition of High End environments at the very outset, we pointed out that automated system administration and operator functions were among the key requirements for many customers. In this section, we will focus on that aspect.

One essential ingredient of automation is programmatic access. Sys admin and operator commands should be programmatically executable, and the results should be programmatically accessible as should the exceptions.

Working from that simplistic definition, the basic issue is to get away from purely interactive interfaces. The ALLBASE team is currently working on a ***programmatic system administration intrinsic interface.*** The specifications were generated early in 1993. The exact availability dates will be announced in the near future. The exact functionality of this intrinsic

interface is as follows. It will provide every command currently executable through SQLUTIL and some additional features. Most notably, ***these intrinsics totally coexist with programmatic SQL commands or SQLX intrinsic calls.*** They are able to share the same session or transaction context of the running application.

The main benefit of the intrinsic interface is that it opens up the ALLBASE environment for automated system management. We also strongly believe that it will make ALLBASE a stronger contender in production shops with a data centre orientation. The term commonly used in such shops is ***lights-out system management***, and this programmatic interface enables that.

Moving on to the subject of performance monitoring, several ALLBASE environments have been successfully using the ***ALLBASE Performance Monitor*** for identifying performance hotspots. While the performance monitor enables users to look at a great variety of snapshots, high end environments will find its ***lock/latch display screens*** and the new ***deadlock matrix display*** feature particularly useful. With a large number of sessions and a large amount of application activity, the key benefit of the Performance Monitor is the speed and friendliness with which it allows DBAs to hone in on the problem areas.

And finally, the ALLBASE product comes with ***SQLCHECK - the database integrity verifier tool.*** Physical integrity is always a difficult subject to talk about in vendor presentations. It is something that the ALLBASE team is proud of, and yet, there should always be a tool available to periodically verify the total goodness of the database. The tool can be run against the entire database environment OR against a set of DBEFilesets. The fileset-level granularity of SQLCHECK enables a High End environment to potentially break up the integrity verification into multiple steps/sessions.

The ALLBASE team continues to work on new features that facilitate system administration in general. Equally important for us are our liaison with 3rd party solution providers in this arena. The organization is constantly looking out for new partners to enhance the overall strength of solutions based on the ALLBASE engine.

6.0 CLOSING REMARKS

It is the author's sincere hope that this paper has been helpful to the ALLBASE-familiar as well as the ALLBASE-unfamiliar class of readers. The product has evolved into an industry-strength relational DBMS engine that is now a strong contender for the High End Marketplace. As an organization, we hope that this paper has interested the reader enough to seek out more information about ALLBASE through their HP contacts.

Client/Server Application Design using Graphical User Interfaces and Distributed Object Technology

#6008
Tim Ryan
Hewlett Packard Company
2205 East Empire Street
Bloomington, Illinois 61704
(309) 662-9411

Use of distributed object technology (DOT) brings the ease of development familiar to Windows desktop applications developers to the client/server arena. DOT provides the capability to create object-oriented client/server applications. Graphical User Interfaces (GUIs) connected to Object Data Bases with network technologies like the Object Management Group's Common Object Request Broker Architecture permit the creation of simple, sharable objects ranging from text to voice and video. These simple objects can be flexibly grouped into end-user definable compound objects that can be shared by far-flung work groups.

This discussion will provide you a framework for understanding how desktop GUIs can be extended to provide flexible, powerful environments for work groups. Through examples, you will see how business applications can be developed to exploit the distributed object technologies that will be available soon from most of the major systems vendors.

INTRODUCTION

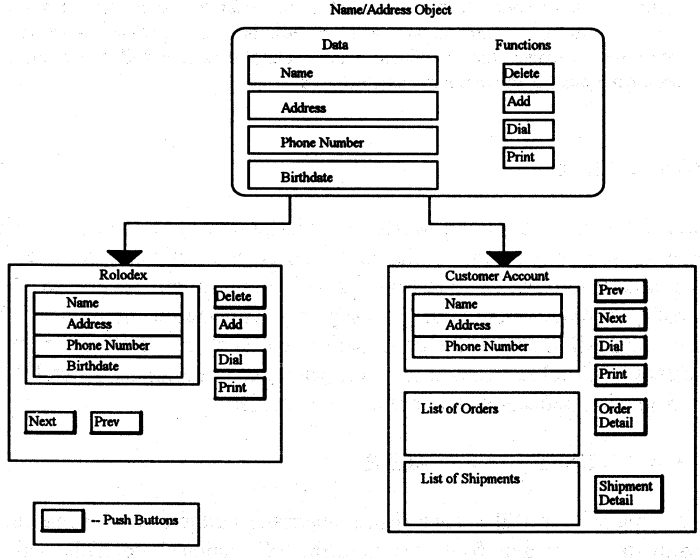
Graphical User Interfaces are powerful tools in the hands of developers who need to create flexible, easy to use applications. These applications become even more valuable and useful when they can provide transparent access to data and functionality spread throughout a company's application networks. In this discussion, we will examine the contribution that Distributed Object Technology, in concert with Graphical User Interfaces, can make in designing powerful, robust, easy to maintain applications that are also easy to use.

GRAPHICAL USER INTERFACES

There are many powerful graphical user interface builders on the market today. They provide everything from a rich array of graphical controls and display formats to a powerful set of visual programming tools to simplify the development of easy to use graphical applications. Much of the power of these tools comes

from their use of object technology. Graphical controls and windows are generally objects of one form or another. A Microsoft Windows (TM) edit window, for example, contains not only a data structure to hold the text, but powerful, editor-like functionality that permits highlighting, word-wrap, insert/delete, mouse usage, etc. This combination of data storage and functionality is a form of object orientation. To take this a step further, if a developer wishes to reuse the functionality of the edit window and enhance it, say to provide tabs, the developer can "sub-class" the edit window and pre-process the tab key, passing the other keystrokes directly into the edit window procedure.

More advanced reuse is possible. In ParcPlace VisualWorks Smalltalk (TM) for example, the developer can create a simple window object or "view" that can then be reused in various applications. This view can be a composite object, consisting of many individual components, both data and function. A name/address/phone# view, for example, can be created that can then be reused in any number of applications from rolodexes to mainline business applications. This name/address/phone# view can contain basic field validations, simplifying the work of the reuser. In addition, the view can be designed to only show certain fields in a given case. For example, the name/address/phone# view can have a switch, that, when set, turns off the display of the birthdate field. This property of objects is demonstrated in Example 1.

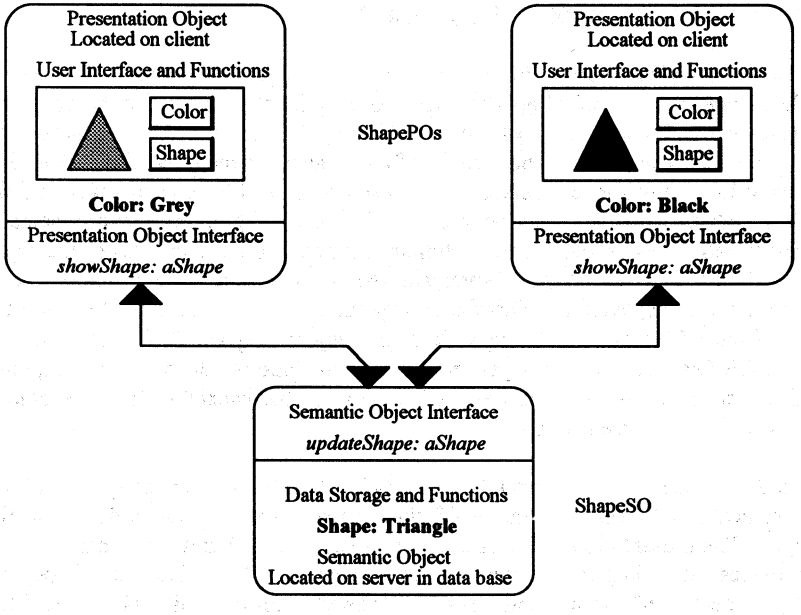


Example 1: Reuse of a name/address "view" in more complex objects

DISTRIBUTED OBJECT TECHNOLOGY

These examples merely hint at the power of object technology to improve development of graphical applications. An additional problem for the corporate developer is the connection of such graphical applications via LANs to data bases on servers. These servers may be mainframes, minis or micro servers. The "data bases" may be relational, hierarchical, indexed sequential files, or just plain old flat files. Much work has been done to provide links of graphical environments to server data bases. Most of these solutions, however, result in the loss of the object orientation that, as shown above, aids so much in the easy development of sophisticated applications. Distributed object technology, such as that provided in the Object Management Group's Common Object Request Broker Architecture (OMG/CORBA) and the Open Software Foundation's Distributed Computing Environment (OSF/DCE) is designed to provide the capability to reuse objects across the enterprise network.

Products such as HP's Distributed Smalltalk (HPDS) build on the CORBA/DCE specifications to enable creation of objects that have a "presentation/semantic" split. The presentation object manages the user interface and the semantic object provides the implementation of the object. For data base objects, "implementation" means "how the data is stored". Products like HP's OpenODB object data base provide transactional integrity and sophisticated query capabilities for stored objects. This presentation/semantic split permits an object stored on a server to have one or more presentations or views on one or more clients. An example is shown in Example 2.



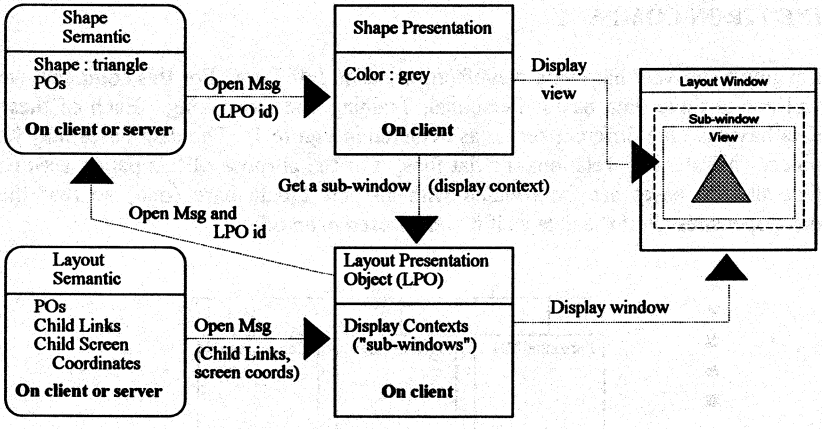
Example 2: Semantic Object linked to Presentation Objects
Shape information stored in Semantic. Color information stored in Presentations

In the chart, a "Shape" semantic object (Shape SO) stored on a server retains the information related to the shape of the object displayed. The color of the shape is stored in the Shape presentation objects (Shape POs). This means that if one user changes the shape, the shape changes in the semantic object and therefore in the presentation objects as well. If a user changes the color in a given presentation object, only the color for that presentation object is changed since the semantic doesn't contain that value. This example is an actual sample application that is distributed as part of HP Distributed Smalltalk. This kind of technology also allows for a semantic object to have links to other semantic objects. For the rest of this paper, I will group the CORBA/DCE specifications and the capabilities of the HPDS and OpenODB products into the category of Distributed Object Technology (DOT).

The presentation/semantic split also provides for the creation and sharing of distributed compound documents. Compound documents (which are in the category of "container" objects) are supported in a windowing environment by a technique known as "clipping". That is, a compound document provides sub-windows known as "clipping regions" or "display contexts" for other objects to display themselves in. The simple objects are passed handles to these sub-windows

and use the sub-windows to display information in. Using DOT, this process may occur over a network.

For example, a particular spreadsheet graph may be created by a user and shared by a workgroup. The graph may then be embedded by a second workgroup user within a word processing document. The graph then appears as a sub-window in the word processing document. If the graph is updated by the first user, the second user's word processing document would be automatically updated. As another example, the aforementioned Shape objects could be embedded in a compound document such as a "Layout" object (a Layout object is kind of a pasteboard for simple objects. It is in the category of container objects.) If the Shape object is updated (for example, the shape is changed) it will change in the Layout object also. The relationship between the Shape and Layout objects are depicted in Example 3.



Example 3: Shape object embedded in Layout object. Layout Presentation Object provides a sub-window (display context) for the Shape Presentation Object to display itself in.

When the Layout object is opened, the following steps occur. The Layout semantic sends a message to the Layout presentation telling it to open. The Layout SO also passes any child link and child screen coordinate information to the PO. The PO uses the screen coordinates to create sub-windows for the child objects to display their contents in. The PO then opens a window on the display device. The PO then sends an open message to any child (or "contained") objects. In this case, only a Shape object is a child. An open message is sent to the appropriate Shape SO. A handle (LPO id) is also sent along with the open message. The Shape SO receives the message and passes it on to the Shape PO. The Shape PO uses the LPO id handle to access the Layout Presentation object and request a display context (sub-window). A handle to the appropriate sub-

window (display context) is returned to the Shape PO. The Shape PO then displays a view of itself in the given sub-window. Please note that this can be a "hot-link", where changing the shape or color of the Shape object immediately changes the view shown in the Layout object.

To describe how applications could be designed with the capabilities of DOT, we'll layout the design of several simple objects and then combine them to create more complex objects. Many discussions of object technology deal with spreadsheets, graphs, video and voice data. While these types of discussions are useful, they obscure the significant gain to be made by using object technology in applications that are primarily text based. This discussion, therefore, will deal specifically with textual information (which is still the primary type of information in business applications).

INFOTRON COMPANY

Imagine a software/hardware manufacturer called InfoTron. For this company, we will create three data bases: Personnel, Training and Consulting. Each of these data bases is on a different server as depicted in Figure 1. The data bases may be object, hierarchical, relational or flat files. For the purpose of this paper, assume that all data bases are front-ended with an object data base (odb) so that the developer sees all of the data as if it were stored in an odb.

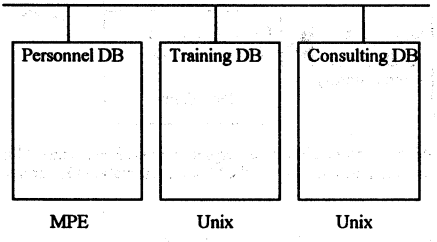


Figure 1: InfoTron Data Bases

PERSONNEL DATA BASE OBJECTS

InfoTron's Personnel department has created some simple objects that describe employees: Person and Employee. (Note: For the purposes of this discussion, the term "object" is used to mean both "object class" and "object instance".) The Person object simply has Name, Address, Phone Number, Birthdate and Marital Status as its attributes. The Employee object is a subclass of Person and therefore inherits all of its behavior. The symbol " :: " used in the figure shows inheritance.

The Employee object also contains Title, Work Address, Work Phone Number, Department and Hire Date (Figure 2). Each of these objects has a semantic part and a presentation part. The semantic part is stored in an object data base on the Personnel Department's MPE server. The presentation part can be resident on Windows, Unix, or OS/2 clients.

The events that occur when an Employee object is activated are as follows. The Employee Semantic Object (SO) on the Personnel DB server activates an Employee Presentation Object (PO) on the requesting client. The Employee SO inherits functionality from the Person SO while the Employee PO provides a sub-window for the Person PO to display itself in. The messages sent by the Employee SO to open the Employee PO also contain the data needed for display. The Employee PO opens on the client display, using the window formats it stores as well as the data passed to it by the Employee SO.

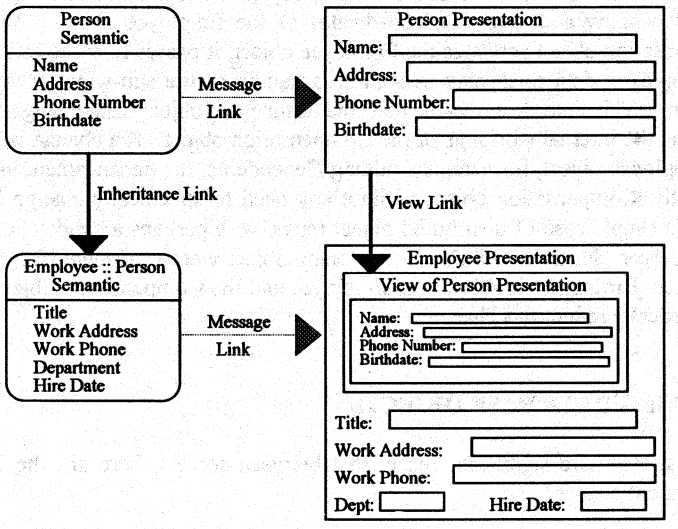


Figure 2: Person Semantic Object is inherited by Employee Semantic Object. Person Semantic Object sends messages to Person Presentation Object. Employee Presentation Object uses a view of Person Presentation Object.

Personnel has also defined a Compensation Object. It contains the Employee object, plus Pay Rate, Bonus Rate, and Job Class. Please note that since a view of a presentation object might be too large to fit into the window provided for it by the other presentation object, the view may include both horizontal and vertical scroll bars. In addition, the Employee view may display only a portion of the data available to the Employee object.

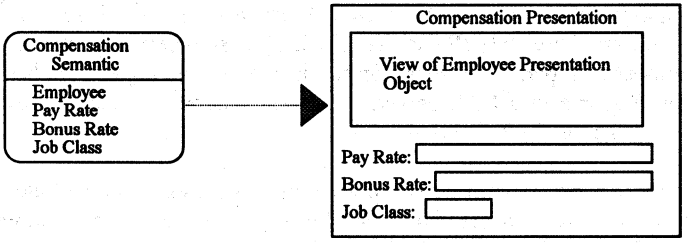


Figure 3: Compensation Semantic and Presentation Objects

The Compensation object demonstrates reuse of a distributed enterprise object. Both semantic and presentation Employee objects are reused by the Compensation semantic and presentation objects. This is different from the Person/Employee inheritance relationship. The Compensation object in effect provides a region in its display window and gives the coordinates to the Employee object. When the Compensation object activates the Employee object, it passes to it the reference to the Employee data to display as well as a handle to the sub-window to display itself in. This interface means that the Employee object can change without affecting the internal workings of the Compensation object. If a change is made to the Employee object, for example, adding Dependents, the enhancement will show up on the Compensation object without any need to specifically change it. This shows a simple case of distributed object reuse, with perhaps a single client and a single server. Note as well that if the Person object were to change (for example, to contain Birth Location) both the Employee and the Compensation objects could automatically reflect this change.

TRAINING DATA BASE OBJECTS

To illustrate more significant reuse, that between servers, here are the Training Department classes.

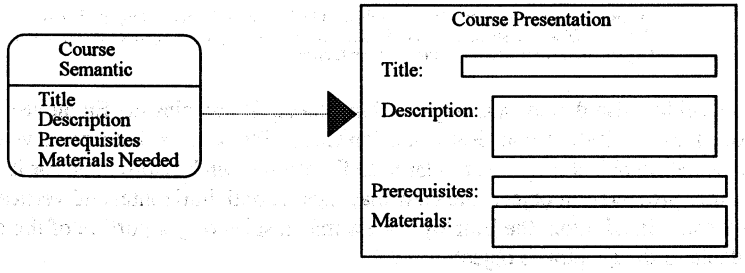


Figure 4: Training Course Semantic and Presentation Objects

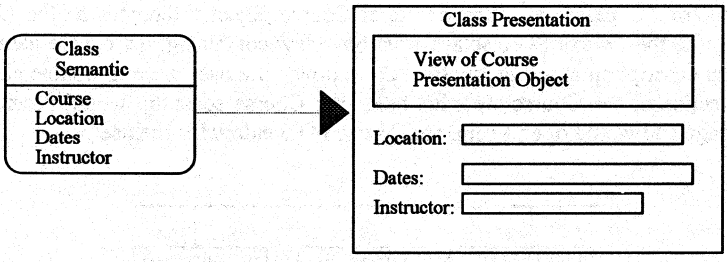


Figure 5: Training Class Semantic and Presentation Objects

Figures 4 and 5 show the Course and Class objects. The Class SO contains a Course SO reference and the Class PO contains a view of a Course PO. The Instructor PO shown in Figure 6 contains a view of the Employee PO. This means that the Instructor SO must contain a reference to the Employee SO. Note that since the Employee SO is stored on a separate server, there must be a network object link from the Instructor semantic in the Training data base. This is depicted in Figure 7.

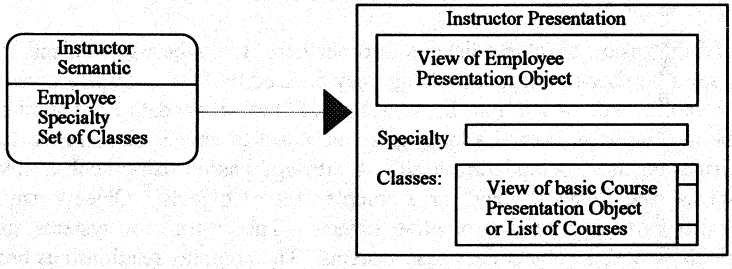


Figure 6: Training Instructor Semantic and Presentation Objects

When the Instructor SO is opened, the object's reference (or "link") to the associated Employee SO is accessed. The Instructor SO sends a message to the Employee SO, telling it to open itself as a view in the Instructor PO. The Employee SO sends a message to the Employee PO, which then opens itself as a view within the Instructor PO. The Instructor PO also displays either a basic view of the Course object or a list box view of Course objects. This view is changed dynamically by the Course object. When a view is presented, if a reference to a single object is passed back to the Course SO by the Instructor SO, the Course semantic displays a single Course PO view in the Instructor PO window. If the

Instructor SO passes an array or list of Course object references to the Course semantic, the Course SO displays a list box view containing just course name and partial description in the Instructor PO window. If a user were to double click on a selection in the Course view list box, the Course semantic would receive the message and would open a separate Course PO window for the user.

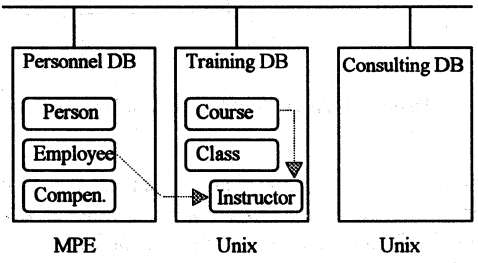


Figure 7: Instructor Semantic Linked To Employee and Course Semantics

OBJECT LINKS

A brief digression on object links is in order here. Links between objects can be used for a number of purposes. A link may be used by SOs to display a view of a PO in another PO. A link may be used by an SO to access data from another SO. A link may be used to send a message to an object to have it perform work, such as displaying itself or updating itself. A message passed using such a link may contain a single simple object or a complex set of objects. Objects may have several links to any number of other objects. This is true for systems such as Smalltalk as well as object data base systems. The complex relationships between different types of data in the real world are difficult to model in a simple relational (tabular) or hierarchical model. Storage with an object model that can more easily resemble the real world greatly simplifies the developer's task in creating robust and flexible applications.

CONSULTING DATA BASE OBJECTS

The final objects to look at are the ones stored in the Consulting data base.

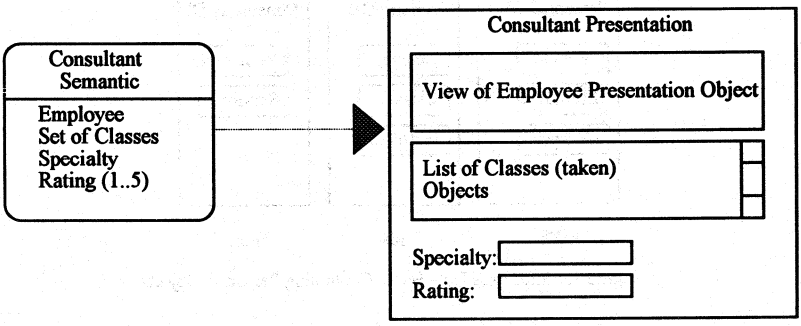


Figure 8: Consultant Semantic and Presentation Objects

The Consultant and Consulting Service objects complete the data base picture. The Consultant semantic contains an Employee object and a set of Class objects and the Consultant presentation contains a view of the Employee object and a list view of Class objects (showing the training that the Consultant has had). The Service semantic contains a textual Service Product description and a Dictionary object that consists of Rating/Price entries. This means that for a higher Consultant rating, a higher hourly price is charged for a particular service. The Service SO also contains sets with descriptions of related hardware and software. This is because the service may come with hardware and software products bundled in.

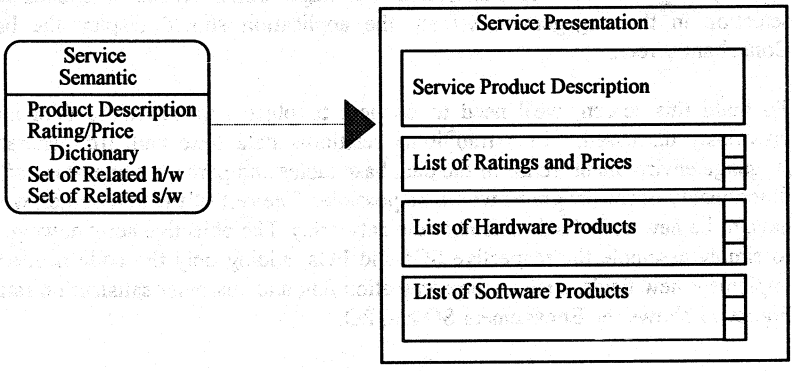


Figure 9: Service Semantic and Presentation Objects

InfoTron's databases now look like this:

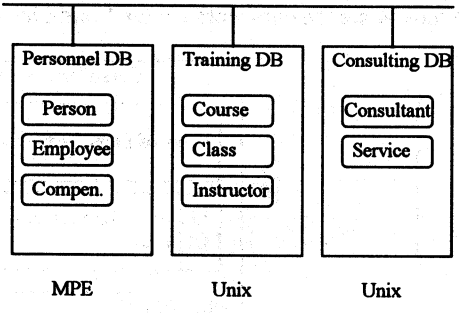


Figure 10: InfoTron's Data Bases Containing Semantic Objects

Not shown in Figure 10 are the links between the various objects.

SYSTEM ENHANCEMENT: ENGAGEMENT OBJECTS

Now assume that a user request for a new type of object (screen) is received. The Consultant Managers want an "Engagement Screen". A "Consulting Engagement" is a combination of Consultants providing a combination of Services to a Customer during a certain period of time. Therefore, for a given consulting engagement, the screen should provide information on the customer, a list of the consultants involved, a list of the services provided, milestone dates for the engagement, as well as a completion flag and a customer satisfaction rating. The request also indicates that "hyperlinks" between the new screen and other screens (presentation objects) be provided. For example, if a manager double clicks on a consultant selection in the engagement screen, the application should display the basic Consultant screen.

To build this screen, we'll need to connect to objects in all of the data bases previously discussed. In a traditional relational data base and 3rd generation language environment, reuse of the data base tables and perhaps some reuse of the functionality of the programs would be possible. The rest of the new screen would have to be new code (and remember the network.) The objective here, however, is to simply assemble the respective SOs and POs, adding only the code needed to implement new fields, such as the completion flag and customer satisfaction rating. Figure 11 shows the Engagement SO and PO.

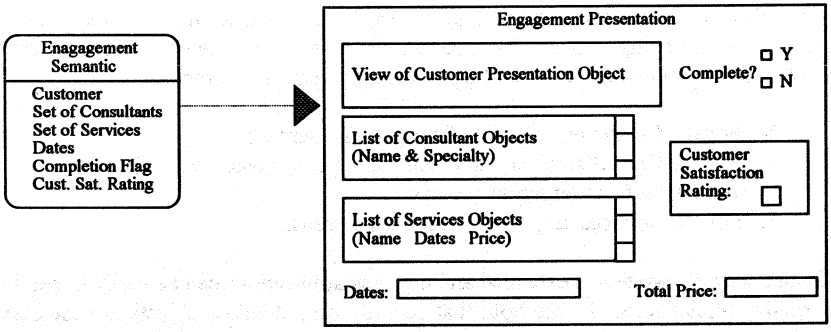


Figure 11: Consultant Semantic and Presentation Objects

In the Engagement SO, we merely define one single (Customer) and two sets (Consultants and Services) of objects that have been previously defined, in addition to defining a few new objects. In the Engagement PO, we reuse a view of the Customer PO as well as list views of Consultant and Services objects. The Total Price field is calculated by adding the individual prices of the Services objects. The "hyperlinks" requested are automatically implemented via the links present in the views provided by the view objects. By double clicking on a given entry in a list box, the user is able to bring up a detail screen on that selection. For example, if a user were to select a Consultant in the Consultant list box view, the Consultant semantic would receive the message and open a Consultant presentation on that object.

HYPERLINKS

These "hyperlinks" could be extended further. Each basic object can inquire as to all other objects that contain links to it. Assume, then, that the base semantic and presentation classes (from which all SOs and POs inherit functionality) have been enhanced to allow objects to determine what links have been made to them by other objects. For example, the Consultant object would be able to determine that it has been connected to Engagement and Compensation objects. It could then provide a pulldown menu that would allow a user to select one of those other objects. For example, given a basic Consultant PO, the user could click the right mouse button on the window. A pulldown menu would appear that listed all of the types of connections available from the Consultant object. Using the above data bases, the pulldown would have Engagements and Compensation Detail on it. If the user then clicked on the Compensation item, the Consultant object would pass the Consultant object identifier to the Compensation object and the appropriate Compensation PO would be displayed in a new window.

This idea can be extended to the Engagement Screen enhancement. If a user were to display an Engagement PO for a particular engagement, the user would be able to access the following from that Engagement presentation screen:

1. For the Customer, any other Engagement screens.
2. For the Consultants, any other Engagement screens, related Consultant screens or Compensation screens.
3. For the Services, any related Services screens.

In this way, the system is extended and any new applications can be easily linked to existing applications. Please note that security on individual objects is necessary since not everyone should be permitted to see the Compensation screen. This type of security can be provided by object data bases. Assuming that basic objects have been hyperlink enabled, the Engagement PO might look like Figure 12.

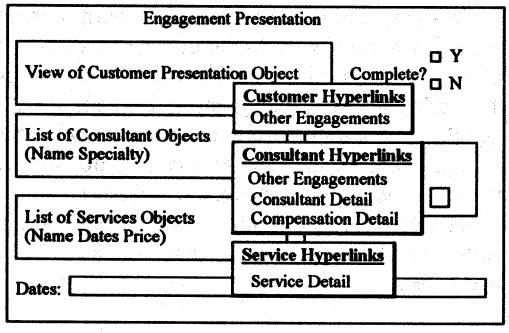


Figure 12: Consultant Presentation Object with Pulldowns for Hyperlinks to related objects

FUTURE CAPABILITIES

Given the above capabilities, the future of DOT will bring the following:

1. Distributed Object "painters", similar to today's GUI painters. This type of painter would generate all of the Smalltalk, C++, ODB, and CORBA language necessary to create a distributed object application. These painters will use business-specific objects created by corporations as well as objects created by vendors. They will use simple, point and click programming interfaces that even end-users will be able to take great advantage of.

2. Object-oriented, distributed modeling tools that will support a high level end-user-oriented perspective of business problems. They will generate object "code" incorporating many of the business rules and business data relationships

(combined data flow diagrams and entity-relationship models) inherent in business applications. Use of object-oriented modeling combined with object language/database implementation retains the unified data/process model from the beginning of the development cycle through to the end simplifying maintenance. It also helps solve the problem of doing high level modeling of distributed (non-mainframe-centric) systems.

CONCLUSION

Object technology (OT) simplifies the development of graphical applications. These applications do not need to be graphs, spreadsheets, or voice and video to benefit from graphical displays and object technology. Simple text objects can be associated and manipulated using OT and GUIs in ways that are difficult to achieve with other technologies. Use of Distributed Object Technology extends this ease of development and use to network based applications. DOT hides the complexity of the network from the developer and end user and permits sharing and interaction of objects between multiple clients with multiple servers. Through the use of object interfaces, links and views, DOT enables the creation of complex, easy to enhance, distributed applications that are also easy to control and understand. Legacy data bases can be integrated in the DOT environment and made easier to access and use.

Experience has shown that hiding network complexity from developers and users significantly improves productivity. Studies have shown that improving reuse provides greatly improved software development productivity. Studies have further shown that object technology provides substantial benefit in the area of reuse. Object technology also enables creation of easy to build, easy to use graphical user interfaces.

Distributed Object Technology integrated with Graphical User Interfaces will speed the development of robust applications. Independent Software Vendors, Corporate Developers, and End User all will benefit from the availability of this enabling technology.

Paper No. 6009

Integrated Imaging In Financial Systems

Adam Thier
Computron Technologies Corporation
301 Route 17 North
Rutherford, NJ 07070
(201) 935-3400

Your organization is being squeezed by competition while at the same time you are trying to capitalize on the new opportunities being presented as fresh markets open up. As a result, you are probably reviewing and revising your business strategies and the information management systems that support them. The challenge you face is to provide more detailed information, more quickly -- while simultaneously improving productivity so you can accomplish this without adding staff.

In other words, you have to do more with less, and quite simply, the profitability of your organization is dependent on your ability to provide new types of financial information on which strategic business decisions can be made. And the only way to do this is by improving the efficiency of the tools and processes you use to gather it.

To achieve this you will require a new generation of financial software which, in addition to being inherently more powerful, will allow you to automate and continuously reengineer your business processes in order to contain costs, provide new types of information, and increase office productivity.

THE FUTURE OF FINANCIAL SOFTWARE

Your financial software is the backbone of your organization's ability to meet the new challenges of global competition, new world markets, and merger and acquisition activities. Without financial information

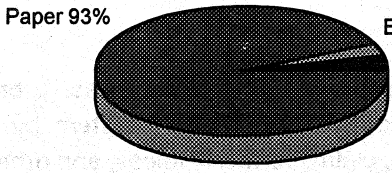
that is timely, accurate, concise, and above all, relevant, even healthy businesses can face financial ruin. An investment in financial software is a significant and critical decision -- and the evaluation criteria of yesterday will not hold up to today's requirements.

In short, the financial software of the future must allow you to continuously satisfy the needs of your users, while providing the flexibility to migrate to your organization's chosen computing strategy -- regardless of its foundations on a mainframe/midrange, or its eventual destination on a distributed client/server architecture built around PC's and workstations. And, most importantly, it must provide a means to automate the office processes that are so heavily dependent on manual efforts and physical devices. This creates true cost-savings and increased productivity, and, for the first time, the ability to measure, control, and refine the office processes.

YOUR BUSINESS ENVIRONMENT

From a business standpoint, financial information is only a portion of the overall information which runs a company. "Data" is no longer limited to just what is in an application's database. For every financial transaction there are an assortment of associated documents and transactions -- word processing documents, incoming correspondence, invoices, E-Mail messages, and even voice mail.

What is startling is that your organization is spending millions on automated data processing when, in fact, only 3% of the information itself is being automated. The rest is on microfiche (4%), and an amazing 93% of all the financial information you process resides on paper. As a result, the information resident in your information system is wholly incomplete.



Bell & Howell Study

Information Storage

Today, the U.S produces 1 billion pages of paper documents daily. And despite information technology, our output of paper documents is doubling every six years.

Quite simply, your organization's information technology investment of the past 20 years -- which is roughly 60% of the money it spent on capital equipment -- has done nothing so far to bring about the promised "paperless office". Instead, it has just given your users the means to produce more paper more quickly.

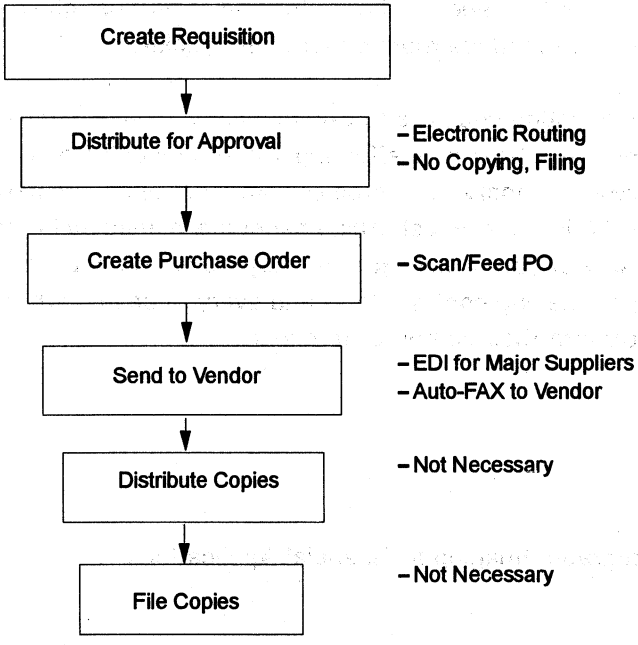
The question you have to ask yourself is what is happening to all this paper, and how is it affecting the productivity of your staff? The answer is unnerving -- studies show that approximately 75% of your staff's time is spent shuffling paper (i.e. managing information). Look closely at your office. Your employees are probably busy referencing paper, making copies, etc. -- and every once in a while -- making an entry into their computer terminals.

INTEGRATED IMAGING

Integrated imaging is the solution to this paper mess. It brings all the information on paper into your information system by taking the purchase orders, receivers, claims forms, invoices, and other hardcopy documents, and turning them into digital computer images via a scanner. It then allows you to display these images on-screen simultaneously with the data represented by your software.

However, more than just displaying images, truly integrated imaging allows you to handle electronic files the same way you handle paper ones. You can file *documents*, in electronic *files* and *folders*, and place them in file *drawers* so they can be retrieved based on different filing schemes. You can add temporary and permanent notes to your files. And you can access images for easy communications via Fax or E-Mail.

Imaging Payment Process



As a result, the simple challenge of finding an invoice when a vendor calls -- which probably now sends one of your clerks or managers scurrying for minutes, or even hours, through mountains of paper, searching for supporting source documents -- can now be verified in seconds by simply bringing the image of the invoice up on-screen. Effectively you regain the 75% of "lost" productivity, and integrated imaging in financial software will provide part of the solution for how you are going to do more without adding staff.

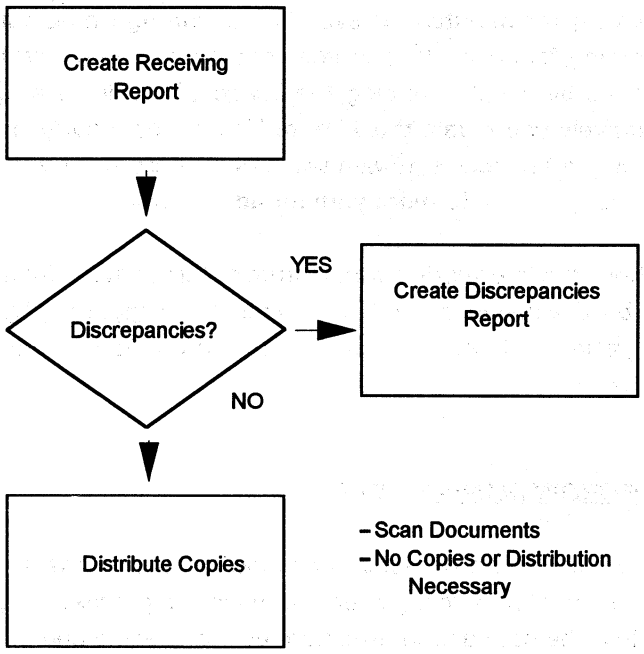
In short, the information architecture of tomorrow's financial software will be developed around the concepts of multi-media -- providing integration and access to all forms of information: data, text, image, and voice.

WORKFLOW MANAGEMENT

There is another process, called workflow management, which allows you to completely reengineer your business processes by automating not just the data and information, but the people and processes who manage the information.

In the past, the single largest part of your department -- your people -- has been ignored by your investments in information technology, and as a result, productivity has remained stagnant. Workflow management systems allow you to define the staff, supervisors, quality control and review personnel, managers, and executives that are involved in any transaction-based process. You indicate their authority levels and how and when information flows from person to person. Workflow management automatically routes information to these people based on your distribution (or routing) rules. Physical in-boxes are replaced with electronic ones, and automated routing rules replace manual copying and distribution. For the first time, your workload is automated, and can be measured. Work can be reassigned to accommodate changing manpower conditions, workloads regulated, and individual performance levels assessed at any time.

Imaging Payment Process



This vastly improves productivity and efficiency. You can do less with more, and gain complete control over not only the information you have -- but who has it, and what they are doing with it. And most importantly, you can easily change the rules determining the flow of work to reach an optimal degree of office efficiency.

Interestingly, and most importantly, workflow can be utilized with or without imaging. In paper-based processing departments, imaging simplifies access to paper files. However, work, in the form of data, without images, still needs to be automatically distributed, balanced, and measured. Therefore, the financial software system of the future will provide workflow management capabilities which can be implemented with or without imaging.

A CASE STUDY

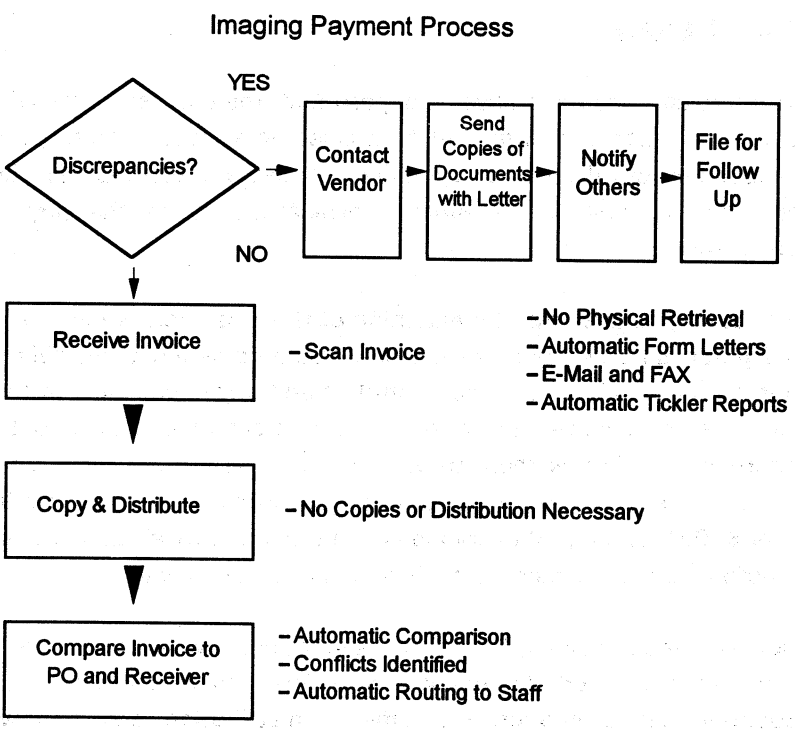
In a "typical" accounts payable department, the process starts with a purchase order, which is sent to a selected vendor. In return, the vendor delivers the items or services requested, accompanied by a receiver, as well as a separate invoice mailed to the payables department.

Processing begins with the matching of the receiver, the invoice, and the purchase order. Most of the time these items are accessible (i.e. not lost or misfiled), and the quantities and prices coincide. In these cases, a clerk makes the appropriate number of copies, and then internally distributes them through the normal routing for payment approval (i.e. purchaser, department head, vice president for invoices over $10,000, etc.). If everyone acts on their approvals in reasonable timeframes the payment itself is generally not delayed.

However, the remaining 5 - 10% of "exceptions", whether they be the result of misplaced documents, or differences in price, terms, or quantity, cause enough problems in distribution and approvals (recontacting the vendor, reapproving new purchase orders, rerouting to other personnel, etc.) that payment is slowed dramatically. The result is, at best, lost discounts, and, at worst, payment of interest penalties.

Further, in both cases, once the invoice/receiver/purchase order packages enter their manual routing processes, their status is almost impossible to quickly ascertain -- there is no way of easily uncovering what in-boxes specific packages are in.

In other words, with manual workflow, the work itself is controlling the way this (and probably your) accounts payable department operates, rather than the other way around -- and workflow management can reverse this situation.



In a workflow integrated accounts payable department, the 5-10% of "problem" invoices, which cause so much lost productivity, become a non-issue -- and the status of all invoices is always readily available.

With integrated workflow, the accounts payable process begins with the scanning of all invoices and receivers as they arrive. These scanned images are then indexed to the appropriate vendors and purchase order files -- and then automatically routed to one or more people for approval. The approval personnel are automatically presented with the work that needs to be approved via their "electronic IN-boxes". For lower-level staff, the work is presented in

order of priority. Upper-levels can browse through their IN-box and choose what to work on; but if they ignore a particular invoice for too long, the workflow system can insist that they deal with it.

If there is a problem with invoices and purchase orders not matching, the workflow system automatically routes the folder to specific personnel for exception processing. Personnel can pop-up a list of standard form letters that are instantly personalized and available for FAX or printing. These letters automatically become part of the electronic file. If a response is pending longer than a pre-defined timeframe, the workflow system takes further routing and notification actions so as to insure that at the very least, desirable discounts are taken, and at worst, payments are made on time and finance charges avoided.

REAL RESULTS

What makes workflow so natural a fit for financial applications is that all companies already have approval processes and routing procedures in place. It is simply a matter of establishing these same "rules" in the graphic routing environment which controls the workflow system.

In the application of workflow technology, productivity increases dramatically while savings are realized. Some typical results of integrated workflow with accounts payable are:

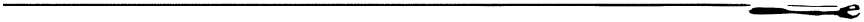
- o 83% reductions in invoice processing time
- o 90% reductions in inquiry response time
- o 77% savings in time required to fulfill correspondence
- o 63% reductions in time required for historical inquiries
- o 95% savings in document storage costs
- o 80% reductions in outbound phone calls

CONCLUSION

The 1990's have ushered in a new business climate. One that provides new opportunities for success....or failure. Those organizations that survive and prosper will be those able to respond quickly and effectively using the information system tools they implement today.

These requirements have set new standards for the evaluation of financial systems. Your financial applications software needs to be able to provide the ease-of-use and user-customization routines that ensure the ability to continuously meet your user's needs. The system must possess an architecture that provides a smooth migration to your chosen, and evolving, computing strategy. And finally, the system must enable your organization to work faster and smarter.

Integrated Imaging and Workflow Management address these productivity issues by combining to provide users with not only immediate, on-line access to all the information crucial to their jobs, but with the means to control and streamline the business processes that revolve around that information.



PAPER NUMBER: 6010

TITLE: Client-Server: The Best Architecture for Business Applications

PRESENTER: Gia Knaus
PeopleSoft, Inc.
1331 N. California Boulevard
Walnut Creek, CA 94596
510-946-9460

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

PAPER NUMBER: 6011

TITLE: Improving the Efficiency and Accuracy of Your Information Capture Through Automated Data Collections

PRESENTER: Ray Agrusti
Eagle Consulting & Development Corp.
170 Kinnelon Road
Suite #3
Kinnelon, NJ 07405
201-838-5006

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper # 6012

Moving to Open Systems with a 4GL

Billy S. Hollis
Zortec, Inc.
1321 Murfreesboro Road
Nashville, TN 37217
(615) 367-6028

Abstract

Downsizing from older proprietary computers to an open systems platform such as the HP9000 can offer many benefits. However, moving existing software systems off the proprietary environment presents a great challenge. Code translators are often ineffective, and rewriting from scratch means throwing away a large investment in existing code.

Fourth generation languages (4GLs) offer another potential solution. Faster software development makes rewriting programs more feasible, and migration/conversion tools allow significant preservation of existing software investment. However, 4GLs solutions may have additional hidden costs, depending on the language chosen and how it fits into your environment.

We will discuss several downsizing strategies involving 4GLs, and examine the advantages and disadvantages of each. A case study of each strategy will be presented for real-world comparison of results. Code conversion, and its limitations, will be examined in detail. We will also discuss how to choose the right 4GL for your migration needs.

We all know a trend has definitely arrived when airline in-flight magazines start doing articles on it. Downsizing has reached that point, and that puts you, as an MIS person, at risk. At any time, a CEO (or, worse, a CFO) may come into your office demanding to know what your plans on downsizing are. By then, you'd better have an answer.

Your installation may have good reasons not to downsize. If so, a clear presentation of those reasons is all you have to do. But you should be careful in making such assumptions unless your reasons are really unassailable. The economics of downsizing can be compelling. Especially if your platform is an older, proprietary computer, an entire downsized replacement may be cheaper than what you pay for one year's maintenance.

So let's assume you have decided to downsize, either now or at some convenient point in the future. That decision is the easy part. Now you have to find a way to move your company's software and operations onto a new platform - in a reasonable time frame, and keeping disruption small enough to stay in business.

The toughest decision you will make is how to move the software that your organization depends on. Your options are bewildering - rewrite from scratch vs. code translation, C vs. COBOL vs. a fourth generation language (4GL), in-house conversion vs. contracted out-sourced conversion. A poorly planned transition to open systems can cost your company far more than the up-front expenses.

In this paper, we will discuss some of the available strategies for moving software to open systems, and examines in detail one of the common choices - using a fourth generation language.

Migrating Existing Code

Often, the early front-runner for a migration solution is to move existing software over with as little change as possible. There are points in favor of this strategy. It preserves investment in existing code, for example. Too often, though, this solution is chosen simply because everyone feels more comfortable with it.

The biggest flaw in this approach is that it perpetuates difficult-to-maintain code. Existing programs may be ten years old, and patched and repatched until hell wouldn't have them. If the current code is a hopeless mess, the migrated version won't be any better. Most shops spend around three-fourths of their time doing maintenance - more if the code is very old. One of the typical reasons to downsize is to provide new functionality in the migrated system, and that is impossible if simple maintenance takes up almost all the available resources.

Migrating existing code is also tougher than it may appear. COBOL compilers usually support the ANSI standard, which basically means they are all different in the same places. Those differences, usually referred to as extensions, mean it is unlikely that your current COBOL will run on other platforms without a large amount of manual tweaking. COBOL translators may be used for some of this effort, but no automatic process can do all the work.

Rewriting From Scratch

Most installations examine the obvious alternative to migration of existing code, which is to rewrite the software from scratch. And that leads to the question of what language to choose.

There are usually three choices that are seriously considered - the installation's current language (typically COBOL), C, and a fourth generation language (4GL).

Since downsizing usually means going to a Unix-based system, many installations first look at rewriting in C. However, it's almost always the worst choice for a corporate MIS shop, and is usually a poor choice for a vertical market software developer. Even many C proponents describe the language as best suited for system-level coding and not recommended for application programming. Here are the basic reasons C is unsuitable for most business-oriented programming:

- Programs written in C are difficult to maintain.
- C has significantly lower programmer productivity.
- C programmers are expensive and in short supply.

C does have some strong points, such as portability and performance. It has a slight speed advantage, usually 10 to 25%, over most business-oriented languages. C source code is reasonably portable, certainly more so than COBOL. But for most business applications, the marginal portability and speed increase do not make up for the disadvantages of C.

Rewriting in COBOL often has appeal because the programmers are already available to do the job. This choice is not as bad as rewriting in C, but there are drawbacks. A complex system can take a long time to create. And the resulting code is not very portable or particularly easy to maintain. There's also

the temptation at every stage to get out that old spaghetti code "because we know it works", which leads back to even worse maintenance problems.

Taking the 4GL Plunge

4GLs have some capabilities that would seem to make them a natural fit for the rewrite situation, so one or more of them are usually evaluated as a migration option. To see why, let's first discuss what a 4GL is.

Understanding the term "fourth generation language" requires a bit of history. The very first computers had to be programmed in binary. The "second generation" of programming was launched when mnemonic assemblers were introduced. A bit later, "third generation languages" such as COBOL and FORTRAN were created. Each generation of computer languages resulted in a big jump in programmer productivity.

In the late seventies, the understanding of computer software had evolved to make possible another leap in programmer productivity. The resulting "fourth generation" languages had the objective of make software development around ten times as fast as COBOL and related languages allowed.

There is no industry standard definition of a 4GL, but here are some characteristics shared by most 4GLs:

- Program sizes are about one tenth the size of an equivalent program written in COBOL.
- The language has built-in database capabilities.
- A data dictionary is used to hold all data structures.
- The language contains an integrated debugger.
- The language has a lot of intelligence built-in. It makes default assumptions about what the programmer wants wherever possible.
- Application generators allow file maintenance and report programs to be written in five to ten minutes.

- A subset of the language can be taught to non-programmers in a two-day training course, giving data access to a wider range of users.

- Results can be obtained in an order of magnitude less time than with COBOL.

Most 4GLs are also quite portable, some all the way down to the level of the compiled code. That allows, for example, exactly the same software to be run on an HP3000 system as on a brand-new HP9000.

From a MIS manager's point of view, the above characteristics translate into three primary advantages to using a 4GL for downsizing:

- Quick development

- Easy maintenance of the resulting code

- Portability to a new open systems platform

Other secondary advantages include giving non-technical users better access to data, and easy adoption of new software technologies such as windowing and client-server architectures.

If 4GLs Are So Great, Why Don't We All Have One?

This is an impressive set of benefits. However, 4GLs have been around awhile and have not achieved overwhelming acceptance. The main areas of dissatisfaction for most 4GL users fall into three catagories

- Poor performance

- Lack of flexibility

- Steep learning curves.

It is not unusual for a fourth generation language to require five to ten times the resources used by COBOL for an equivalent application. This translates to a much higher cost requirement for the replacement open system, and continued higher prices for later expansion.

The flexibility problem stems from the high-level approach taken by most 4GLs. It's easy to write data-entry applications and reports, but the languages may lack the constructs to do lower-level programming. That means a second language (usually COBOL or C) is required for such work. Programmers then have to be trained on both languages, and the resulting multiple-language software is more difficult to maintain.

Steep learning curves are a consequence of the academic orientation of many 4GLs. Ignoring current software conventions, some 4GLs have syntax which is obscure and difficult to understand. Together with the conceptual leap of programming in a 4GL, some installations have reported uncomfortably long ramp-up times.

The good news is that modern 4GLs address these problems. Some 4GLs offer performance equivalent to COBOL. And many 4GLs are now so flexible that their utilities are written in the language itself. While it's still possible to get burned with an inadequate 4GL, taking pains to choose one carefully can ensure a good chance of success.

4GLs are also now available that dramatically shorten the learning curve by providing syntax that is intuitive to COBOL programmers. MOVE statements and IF statements, for example, do not necessarily have to look any different in a 4GL than they do in COBOL. While there is always a "hump" to get past in understanding the philosophy of a 4GL, the right choice can make that hump much smaller. Making the right choice is easy - just look at some sample programs written in the 4GL to see if they make sense to you without a lot of training.

Detailed Strategies and Questions to Answer

There are several ways to implement a downsizing strategy using a 4GL. Let's look at some of the aspects that should be considered.

One of the first questions you must answer is whether or not you intend to incorporate significant changes in the rewritten software. The specifications for the old system are typically used as a starting point. You may choose from a range of options, from making little or no changes in the specs up through adding lots of additional capability. In general, there is a trade-off. It is easier to make changes now, but the more changes and enhancements that are made, the more difficult the rewrite will be and the longer it will take. Major changes will probably make reformatting the data more difficult, for example.

There are no universal guidelines for your decision, but remember that one of the advantages of using the 4GL is fast development. It is much more practical to add enhancements, especially extra reports, with a 4GL than it would be with COBOL.

Another question is less obvious - what platform should be used for the development? The simplest approach is to acquire the new, downsized platform along with the 4GL to be used, and just turn the programmers loose on that. There's nothing wrong with this, but another alternative is worth considering. If the 4GL you choose is portable to your old platform, it may be more practical to do the rewrite there. This allows you to put off the purchase of the new hardware until the software development is done, and relieves the programmers of adapting to new hardware and new software simultaneously. It may also make it easier to test your new system in parallel with the old one.

Software Conversion

Some 4GLs offer migration and conversion tools that may affect your strategy. At the simplest level, tools are available to build the 4GL's data dictionary from the file definitions already present in your source code. This can save lots of manual work, and make the new files more consistent with the old ones. Depending on the 4GL, you may even be able to have new programs read and write to the same files used by your existing software.

Some conversion tools take code in your present language and convert it to 4GL code. This offers you the prospect of getting your migration done more quickly, but there are tradeoffs.

As earlier mentioned, no conversion or translation tool is one hundred percent effective - some manual work is guaranteed to be needed. And the converted code may not be as efficient or as easy to maintain as code written from scratch.

Another surprising factor is that a rewrite may be faster for many of your programs. Since 4GLs offer very easy creation of file maintenance programs and reports using application generators, converting those programs is usually not the best choice.

But for programs that are very complex, and do not need enhancements, conversion can be much better than a rewrite. A complex posting program, for example, may be easier to convert than to write from scratch.

Another instance in which code conversion may be a better choice is when a premium is placed on the look and feel of the code. Some organizations have a goal of keeping retraining of staff to the absolute minimum. Converted code is more likely to look like the system it is replacing than code written from scratch or developed with application generators.

However, if these considerations apply to you, make sure you have examined the customizability of the application generators completely. Better application generators can be set with a template to reflect you choices on user interface, and can often mimic an older standard quite closely.

Related Approaches

Some 4GLs allow a hybrid approach by supporting COBOL syntax "in line". If the 4GL you choose allows this, you may be able to extract complicated logic from an old program, and simply copy it into the appropriate place in the new program. This offers another way to preserve some of the investment in your current code, without going through the effort of a full code conversion.

Combining Techniques

The typical strategy for using 4GL conversion tools goes something like this: Rewrite file maintenance programs and reports (60-70% of typical systems) using applications generators. Then convert most of the rest. Any processing programs that are particularly compute intensive and used a lot can be rewritten for efficiency. And a few programs may lend themselves to a cut-and-paste hybrid approach.

The end result of a 4GL migration can be quite impressive. Many installations report complete, successful migrations in as little as three to four months. Six to twelve months is typical, compared to two to three years with a full rewrite, even in a 4GL.

Getting Additional Help

You should also be aware that a few 4GL companies offer migration and conversion services. This gives you access to personnel trained in the new language and the migration tools. Usually, migration services are custom-designed to fit your situation. You can choose from any level of services. At the low end, you might want one or two vendor personnel to stay on-site through the migration to assist your people. At the high end, you could

contract for a full turnkey migration to be performed completely by the vendor. Your only involvement in that case would be explaining your current software, and deciding on any enhancements to include.

Summary

Using a 4GL as part of your downsizing strategy is much more practical than many MIS managers realize. The reputation of 4GLs, based on early implementations, says that they are resource hogs, limited in flexibility, and difficult to learn. While some 4GLs still have one or more of these characteristics, a careful choice can get you a 4GL which avoids these drawbacks while maintaining the high programmer productivity and software portability you need to accomplish a successful migration.

Code conversion is also an option that can make the 4GL approach appealing. While its applicability varies with respect to individual installations, such a strategy can dramatically trim total migration timeframes.

Information Integration - Issues and Approaches

Frank Quemada
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View ,CA 94043
Phone: (415) 691-5274
Fax: (415) 691-5485

Introduction

As companies move towards adopting open, client/server computing, a natural first step is to use these technologies to provide or create more value out of existing information resources. The focus of information integration is on the integration of existing applications and data for business advantage. It is an activity that combines and "transforms" systems that are currently in place, without substantial application rewriting or re-engineering.

This paper will discuss the problem of information integration, supply a framework for thinking about different integration levels and issues associated with integrating existing systems, and provide an overview of different approaches for integration. In order to discuss what information integration is, it would be helpful to examine what circumstances lead to the need for integrating disparate systems.

Decentralized Policy Leads to Incompatible Systems

"Islands of automation" can arise from a historical business policy of distributing the implementation of different systems and applications to the lowest level of the organization. The results of this policy are different applications and hardware in different departments across the company. Now, businesses have identified the value of integrating these different applications for increased business efficiency or for competitive advantage.

By integrating information from multiple inventory applications, Hewlett-Packard was able to save over a million dollars in the first year after implementing the system. HP's Computer Manufacturing (CM) operations are distributed geographically and organizationally . Each operation maintains its own inventory tracking system. Under the old system, new parts were bought unnecessarily since availability of other parts in inventory at different physical locations was not available to a buyer. An HP project team delivered consolidated information to the buyer's desktop by integrating the available inventory in separate applications using client/server technologies. The Inventory Visibility Module (IVM) provides information on parts in inventory companywide. Using the system, a buyer can source parts internally before going to external vendors and find internal buyers for excess inventories. In addition, HP management uses the system to analyze inventory for strategic purposes.[1]

A manufacturer of heavy machinery was able to significantly cut costs in the shop floor data collection area of their manufacturing plant. Previously, salaried employees entered data received from shop floor employees and input this data into four separate systems: payroll, inventory

control, material tracking, and production management. By implementing a new system, one transaction updated all the required systems automatically. This system also saved the costs of the salaried employees since it could be administered by the shop floor employees. The company was able to save $200,000 annually through reduced staff and decreased maintenance expenses.[2]

Need to Integrate Arises from Acquisitions Strategy.

The need to integrate information across the enterprise can come about in other ways. For example, a company may be expanding through acquisition and consequently find itself having to deal with different applications and hardware platforms in order to perform some basic business functions.

A Mid-west company had a company vision to become the industry leader in logistics. In order to achieve that vision it pursued a strategy of increasing market share through acquisition. After 5 years and acquiring 4 companies, the company ended up with an organization that consisted of 5 business units located at 3 different locations. To compound problems, different parts of each business unit resided in each of the locations. To make matters worse, each site had its own accounting system (general ledger) and financial reporting package. The company's chief financial officer had a need to integrate information from 3 different systems in order to perform financial analysis and reporting on each business unit as well as to generate a consolidated general ledger for the company. In this case, integration was needed in order to perform basic business functions as well as financial analysis.[3]

What are the Benefits of Integration ?

The examples above show that companies can realize the value of data that is currently locked in their existing applications. Some benefits that come about by unlocking these treasures are:

- Increased efficiency or time-based competitiveness.
  - Access to enterprise wide data.
  - More timely access to data
  - Automation/elimination of redundant tasks or processes.
- Improved customer satisfaction through improved services.
  - Quicker response to customer requests.
  - Wider range of services.
- Solution to a specific business problem.

The key requirements to obtaining these benefits can be summarized by:

- A need to preserve the investment in legacy applications while incorporating new, open client-server technology.
- Need to integrate without rewriting applications.

- Rapid development of solutions (under 12 months).

Information Integration Defined

As shown in the previous case examples, originally separate systems are linked in such a way that they exchange or share data. Additional integration can be achieved by coordinating different applications so that they perform functions or services for each other. In order to tie all these applications together at the desktop, a common graphical user interface must be supplied to the integrated system. To summarize:

> *Information integration is the linking together of two (or more) information systems that were developed independently so that they exchange data, perform services for each other, and present a consistent, transparent interface to the end-user.*

By consistent is meant a uniform look and feel to the system in the way of a graphical user interface. Transparency implies the access of data regardless of its location in a way that is not noticeable to the end-user. Applications can exchange or share data and cooperate with each other in a variety of ways which will be described later.

Integration Framework

Given the benefits of integration, how does one go about getting started ? First, a business case must be built and opportunities for integration such as those described earlier must be identified and justified. Once the need has been identified, a framework for thinking about integration needs to be defined. This section will provide an overall integration framework and following that, general approaches within this framework will be discussed. Please note that I am not addressing here the design of integrated applications which involves information architecture, data distribution strategies, and how to handle logical and semantic differences in data.

Integration Levels and Issues

The type of integration discussed here involves system design and applications design with regards to data and the way it is modeled, stored, processed and presented to the user. It also involves the design of application processes or functions and how those are distributed across the enterprise.

How this integration is accomplished can be described in terms of :

- Data integration.
- Application integration.
- Presentation integration.

These three information integration levels are listed in the table below and corresponding issues mapped to each category.

Integration Level	Issues
Data	• Database selection. • Gateway availability. • Data distribution: replication, copies, extracts. • Data conversion. • Consistency- distributed transactions. • Database schema design and conflict resolution. • Host connectivity: ▪ Terminal emulation. ▪ Write to Host protocols. ▪ Gateways. ▪ File extracts and transfer.
Application	• Application distribution. • Inter-process communication: remote procedure call, messaging. • Version control. • Security (encryption). • Data conversion. • Global transaction management.
Presentation	• GUI interface design.

In general, data integration focuses on providing an integrated view of the data to an application. This is usually accomplished by providing a common data access language (SQL) and combining, or consolidating database schemas in a shared or global form. Application integration focuses on enabling cooperation between applications for the purposes of control (starting or stopping processes) or sharing of data. This cooperation is accomplished through different messaging services that enable inter-program communication. The third layer- presentation, uses common graphical user interfaces to present data in such a way that the underlying separate applications appear as a seamless whole that supports the particular business function.

Typically, application integration is used when real-time access to data is required (sometimes the data may exist only within the application and not in a database). When data needs to be shared among applications in real-time, various messaging services are used to facilitate inter-process communication. Another need addressed by this area is to coordinate or control different applications- a requirement commonly found on a manufacturing floor.

Data integration is found in traditional decision support applications. The focus is on allowing the end-user to access enterprise-wide data. Real-time access to data is not a critical requirement here. Approaches in this area emphasize providing a common view of data to the application and end-user or providing quick access to data throughout the enterprise.

The rest of this paper will concentrate on approaches for achieving data integration. Two different models, a two-tier and three-tier, will be discussed. These approaches provide different methods for accessing data. The two-tier approach emphasizes a common data access language (i.e. a common view), and a three-tier way uses shared functions or procedures to provide rapid access to data. The issues listed above are applicable whether a two- or three-tier model is planned. As will be shown in the case study, combinations of two- and three-tier methods can be achieved and in cases where the integration issues are more complex, these hybrid solutions will probably be used most often.

Client/Server Architecture and Legacy Applications

After defining an integration framework the next step is to identify appropriate models of client/server computing to use. Two major styles of client/server computing (among others) have been used for integrating legacy data. These models have commonly been described as either a 2-tier or 3-tier client/server architecture.

An application program can be divided into three main logical parts consisting of presentation, application logic, and data management.

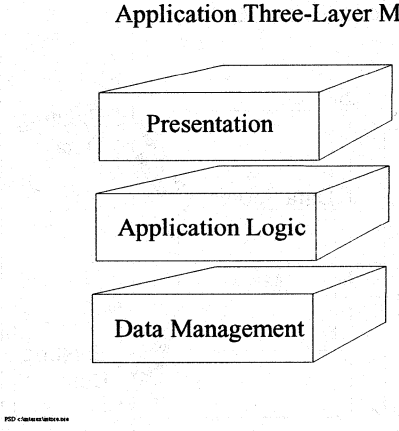


Figure 1

These layers[4] are defined as:

- Presentation: Application component that interacts with an end-user input device such as a terminal, PC or workstation.
- Application: Application component that uses the input data (from an input device or database) to perform business functions and processes.
- Data management: Application component that manages the physical storage and processing of data in any database, file or data store.

Some terms used in this paper:

- Client: A process that asks for services to be done on its behalf. Also used in reference to the computing platform the client process resides on.
- Server: A process that performs services on behalf of a client. Also used in reference to the computing platform the server process resides on.
- Legacy: Existing applications and data.
- Host: An existing computing platform.

Two-Tier Architecture

In a two-tier model the functionality and presentation layers are resident on the client while the data management resides on the server.

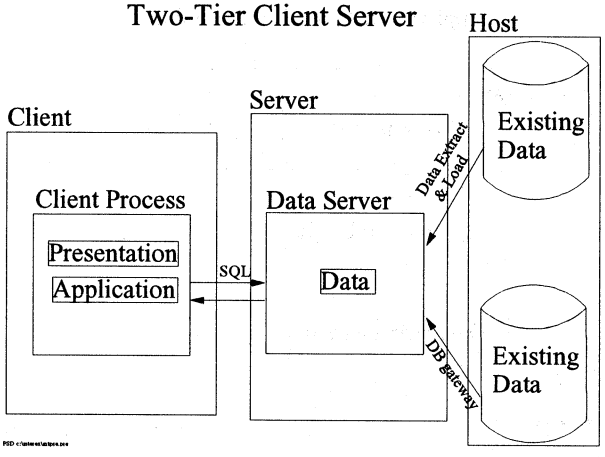


Figure 2

In this example, the client will issue a request in the form of SQL commands which are processed directly by the server and the results returned to the client. Host or legacy data are pre-loaded into the server at specified intervals. In this type of solution, host data is not accessed directly but extracted and loaded into a new data server (typically a relational database). Another option available for accessing legacy data is to use a database gateway which provides real-time access. In the case of a gateway, the connection is handled by the data management layer. The two-tier way is a traditional technique for providing decision support and a graphical interface to existing data. This approach provides easier access to data through a graphical user interface. It is a well proven model for providing robust decision support systems.

The communication between client and server is accomplished through a common API (Application Programming Interface) using SQL. These API's can be supplied by the database vendor or by software language vendors and reside on both the client and server. As a result, choice of clients is limited by the number of API's the database software supports. In turn, it would be difficult to replace the database server without having to rewrite or replace the client since the data access specific to that server is coded in the client. Clients typically use a particular database's flavor of SQL, which inhibits easy migration to another database. Since data is loaded at specified intervals, direct on-line access to host systems is not available unless provided through a gateway. Updating legacy data is usually not performed in this case. Some gateways may provide limited capabilities for updating legacy data.

Three-tier architecture

A three-tier model splits out the application logic from the client and assigns it to a specific functionality server. The client has presentation and the server(s) have the application and data layers. In this model, communication is performed among three levels including presentation, application, and data.

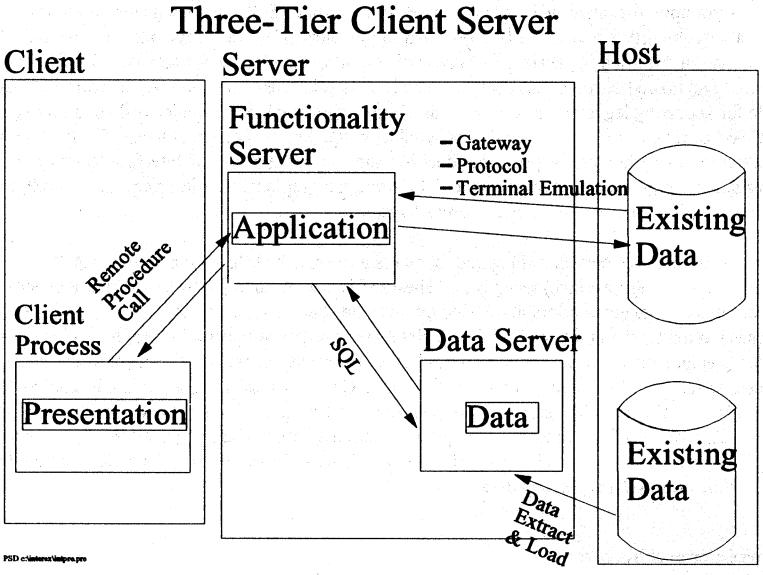


Figure 3

The functionality server sits between the client and the data. It serves to make requests for data from a database or extract data directly from a host system. Consequently, three tiers are involved in fulfilling a particular transaction request. A client issues a request for data which is received by the functionality server, which in turn extracts the data from the data management layer (host or database) and returns it to the client. The functionality server can also extract data from a relational database using SQL. In the case of legacy data, this server can be written to use gateways, terminal emulation (screen scraping) or the protocols necessary to connect to the host system where the legacy data resides. Also note that the functionality server can be implemented on a separate computing platform from the data server.

This model is most commonly implemented using remote procedure calls (RPC) for communication between the functionality server and the client. This allows a desktop client to access different data sources by calling common functions. The peculiarities of access to each source are handled through the specialized functionality servers which can be placed on a more powerful machine. As the number of systems to be integrated increases and the differences in communication protocols and data formats also increase, a 3-tier model is found to be more useful and

flexible . The advantages of this approach are greater flexibility in communication (a greater ability to accommodate different kinds of connectivity to host platforms) and the ability to distribute application logic. The functionality server can be written to directly update legacy data. In this style of computing the client does not communicate directly with the data management layer but through a middle tier of functionality servers.

The addition of a middle tier also allows for greater independence in the choice of database servers and clients. Since the client is not closely linked to the database server, it is easier to change databases without having to rewrite the client. The functionality layer buffers the client from any database changes. A wider range of clients can also be supported than originally provided by the database server since communication with the client is performed by the functionality layer. Different types of clients can access data by calling shared procedures which in turn obtain data on the client's behalf. By using this approach it is possible to distribute application logic across different computing platforms in order to more efficiently match computing resources with application functions. In addition, the use of a well defined API to the data, instead of SQL, promotes re-use for other application developers. An inventory of API's can be developed that provides developers easy access to data in contrast to the alternative of analyzing and interpreting database schemas.

The next step to a 3-tier architecture is distributed computing (multiple tiers) which allows for the distribution of application and data across multiple platforms. This requires having a robust computing infrastructure in place such as DCE (Distributed Computing Environment)

Two-tiered vs. Three-tiered Data Access

Both approaches provide an integrated view of data to the client but through different methods. A two-tier approach provides a uniform view of data through a common data access language (SQL). A three-tier approach provides a common view of data through the use of shared functions which return data requested. In addition, a three-tier technique hides the complexity of data from the client. As the environment becomes more complex and more heterogeneous a three-tier approach provides more flexibility. It allows for easier replacement of clients and servers. Different methods of connectivity (including on-line access) with legacy data can be accomplished. On the other hand, as the scope of the middle tier increases, it becomes more complex to develop and manage.

A two-tier approach provides advantages in environments requiring high performance query over large (gigabytes) databases. But it is also useful for setting up simple database client/server applications where data is batch loaded from legacy applications. For many companies, the two-tier approach is a simple way to start building client/server applications.

Technical Approaches

Within a two- or three-tier model different technical alternatives are available. The following discussion will provide an overview of the different technologies used within these models.

The following techniques will be covered below:

- Two-tier approach (presentation and application on client, data on server or host) which is characterized by providing access to data through a common data access language- SQL.
  - Database-centered approach: This approach uses the database as the primary platform for integration. The database is a central hub responsible for providing access to data on other systems. Host data is extracted and loaded into a database or appropriate gateway software is used for connecting and accessing data on different platforms.
- Three-tier Approach (characterized by access to data through common functions) where presentation resides on client, functionality on a server, and data on server or host.
  - Remote Procedure Call. Greater flexibility is achieved through the use of a middle tier of functionality servers implemented using RPCs.
- Distributed environments(OSF/DCE): An industry standard computing environment that provides services required for distributed, client/server applications. A technology infrastructure designed to provide transparent data access across different platforms. Remote procedure calls are also used in this environment.

Database-centered Approach

Database vendors have accomplished data access across different databases through gateways. This software is responsible for connecting across the network and for translating the foreign data format into the vendors own flavor of SQL. This provides end-users the ability to access data on other systems through their favorite database. Typically the gateways work only with the vendor's database. The end-user accesses this data using a front-end client that connects to the database or a client developed with the database vendors own 4GL. The gateway approach usually provides read only access to data. The data may be saved at an intermediate level but very little transformation is done to it. Ingres is an example of an RDBMS vendor that provides gateways to other data sources such as DB2 or non-relational data. SQLBase from Gupta also provides a DB2 gateway. HP Allbase provides a gateway product with read and write capabilities to DB2 and uses LU6.2 data communication. Most other gateway approaches usually require a TCP/IP connection which is usually not a viable option on an existing MVS system.

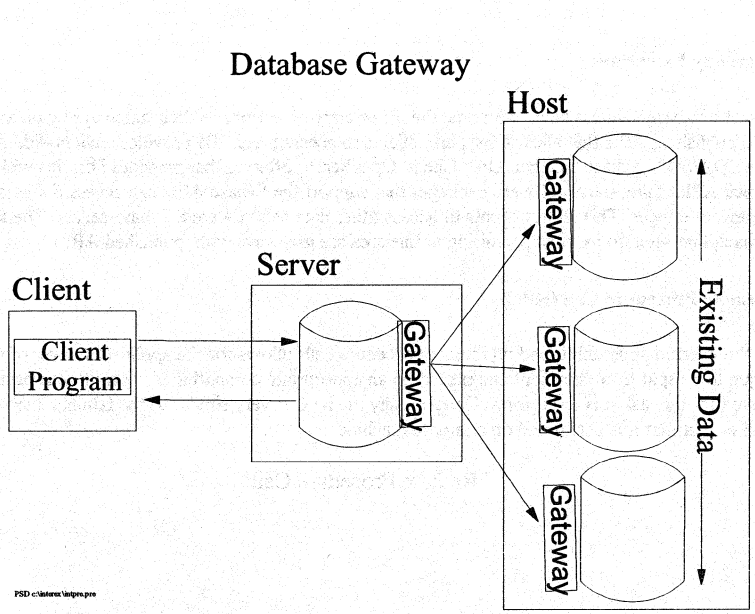


Figure 4

An extension of the database-centered approach is called the "data warehouse" popularized by IBM but originally developed by W.H. Inmon. Data is extracted from the original sources and loaded into a "warehouse" database that provides services to individual departments or end users. This allows for the historical retention of data and also for the transformation of the source data into forms that are more appropriate for analysis and decision support. The underlying assumption of the data warehouse approach is that there are two kinds of information systems: operational and informational. They contain different kinds of data and serve different purposes. Operational data with its focus on current data is not suitable for historical or time-based analysis and must be transformed via the warehouse architecture into a more suitable form. A technology for accomplishing the data warehouse architecture is IBI's EDA/SQL which provides connectivity to over 50 data sources using their own specialized gateways.

The database gateway has been the conventional method for data access and connectivity across networks. Limitations of this approach are the proprietary nature of the gateways, performance, and their limited functionality.

Gateway Extensions

Some database vendors have overcome the proprietary objections to their gateways by providing and publishing APIs that allow third party clients to access them. IBI provides a client-side API for EDA/SQL. Sybase has their OpenClient, OpenServer offering that provides libraries with published APIs. Third party software packages that support the Sybase APIs can access Sybases' gateway services. This allows clients to access other data as if it were a Sybase server. The approach that seem to be most promising in this area are gateways with published API's.

Remote Procedure Call (RPC)

A three-tiered approach based on remote procedure calls allows for the application layer of a program to be split from the client and placed on an appropriate computing platform that is best suited for the task it is to perform. Conceptually the RPC is very similar to the familiar subroutine call except that it is performed on a remote machine.

Remote Procedure Call[5]

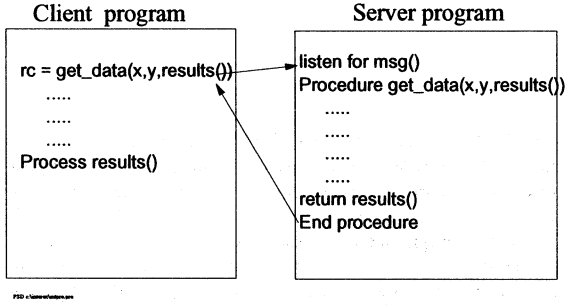


Figure 5

Prior to the call from a client, the server is running and listening on a port for messages from different clients. Upon receipt of the client request, the server performs whatever functions necessary (access to a database, establishing a remote connection to a host etc.) to extract data and return it to the client. A database is not the only means used to access data. The remote procedures reside in special "function" servers that are programs which access host systems directly through:

- Terminal emulation.
- Host platform protocols to connect directly to legacy databases or applications.
- SQL queries. These servers can be written to execute SQL statements and thus provide a "call" level style of SQL versus embedded SQL.
- Direct access to non-SQL databases as well as file systems.

The software components involved in an RPC call include client and server code stubs as well as run time libraries to facilitate connection.

RPC-Based Approach[6]

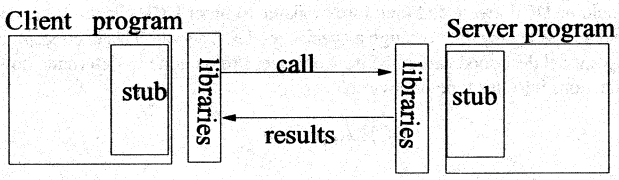


Figure 6

The advantage of this approach is that the complexity of the network protocols is hidden from the programmer. The RPC mechanisms also perform some level of data conversion. On the other hand, servers generally are written in a language such as C and require specialized skills to develop. The most popular RPC software on the market are available from SUN and HP. A development environment is available from CTG-OEC which makes it easy to create Unix servers with PC clients using an RPC based style of communication. Using these tools one can quickly construct three-tiered solutions for integrating data. These tools also provide the ability to develop SQL servers- i.e., SQL statements can be executed against a database via a call level interface.

Open Distributed Environments

This approach does not use a database as the central platform for integration but instead supplies an entire computing environment with services that provide, among other things, network connectivity and data access. These environments provide an infrastructure for distributed computing environments.

OSF/DCE (Distributed Computing Environment)

OSF/DCE provides a robust computing infrastructure for developing and deploying large scale client/server systems. It also provides security (authentication and authorization), naming services, threads, and is RPC based. Applications need to be written in a low level language, typically C, to use these environments. Interprocess communication is accomplished through the RPC mechanism which shields the developer from the complexity of network protocols. Data exchange and translation are also handled via the RPC services. Legacy applications would be difficult to integrate without rewriting. One approach would be to write a front-end server to the legacy application that would be DCE aware and therefore available to other DCE clients. In this case, DCE clients would access legacy data through a specialized DCE server. This is conceptually similar to the three-tier model described earlier. More work needs to be done to determine how to integrate legacy applications into this type of environment.

DCE Approach

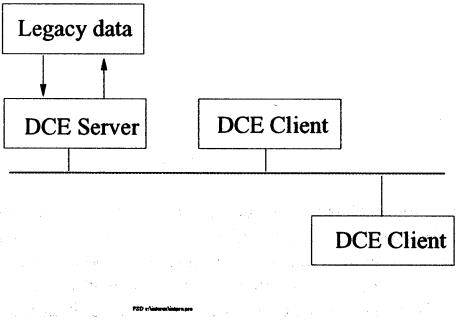


Figure 7

Summary

Legacy data can be brought into a client/server environment without having to rewrite or reengineer your existing applications. Through the use of the models and technologies described,

information integration can provide quick and measurable results in employing client/server technologies for competitive advantage.

Case Study

The following case study focuses on the problem of integrating different inventory tracking systems. It discusses different approaches to the problem using a database-centered approach and shows how, depending on the complexity of the systems to be integrated, a need for application distribution and the 3-tier approach is recommended.

The business case for consolidating parts inventory discussed earlier will be used. Different divisions have implemented different inventory systems. In order to facilitate the internal sourcing of parts and the disposition of excess inventories, information on parts availability world-wide needs to be provided.

Database-centered Approach

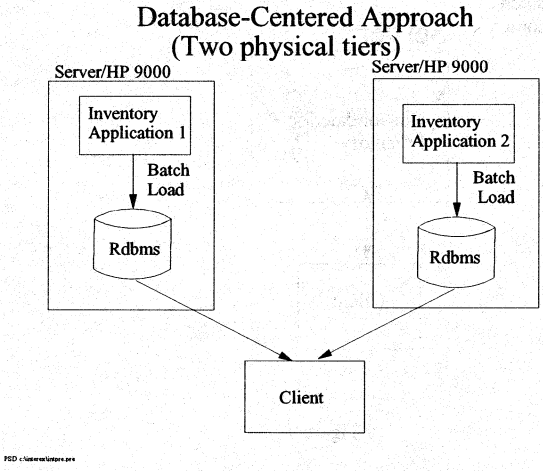


Figure 8

In this approach, you can see a physical implementation done according to 2 or 3 tiers. However, the data access methods in both cases is still primarily two-tiers. Data from the existing inventory systems are batch-loaded into a database that resides on the same platform as the application. The

databases are responsible for translating the inventory data into the integrated view of data as well as any necessary data type and format conversions. The client can be any one of the front-end packages supported by the database vendor. Communication is done using common SQL API's. In this approach, the environments are similar enough to support the same database that acts as the "data integrator." Inventory records for parts are loaded into the local database. The client then issues requests to each database to obtain parts inventory. This method does not provide real-time status on inventory. In this case, management and control of data is retained at the original source sites.

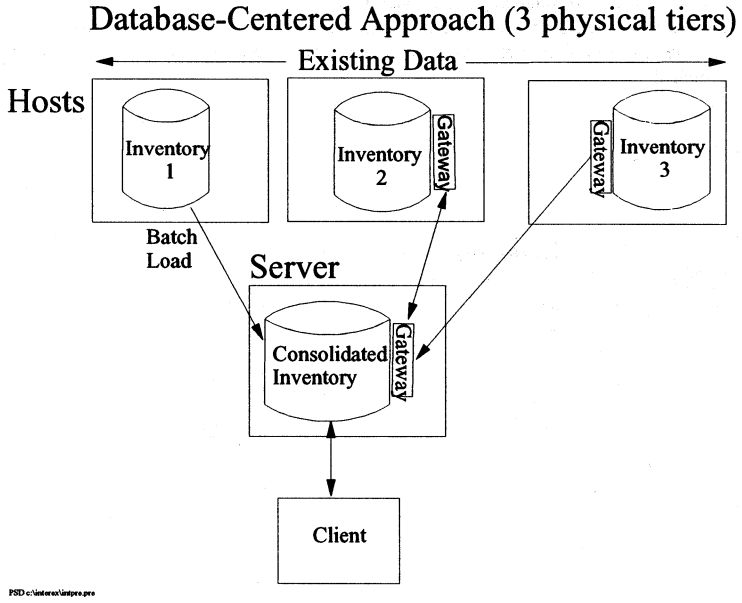


Figure 9

This approach illustrates the use of a gateway to access data as well as the introduction of a middle computing platform that is separate from the platforms where the legacy data resides.

This method uses gateways to access two of the source databases. In this case, the database vendor is able to provide specific gateways that fit this situation. Consequently, real-time information from inventory database 2 and 3 can be obtained. Please note however that for database 1, data is

still loaded in batch. In this case, another issue to address is responsibility for control and management of the data in the middle-tier.

RPC based approach

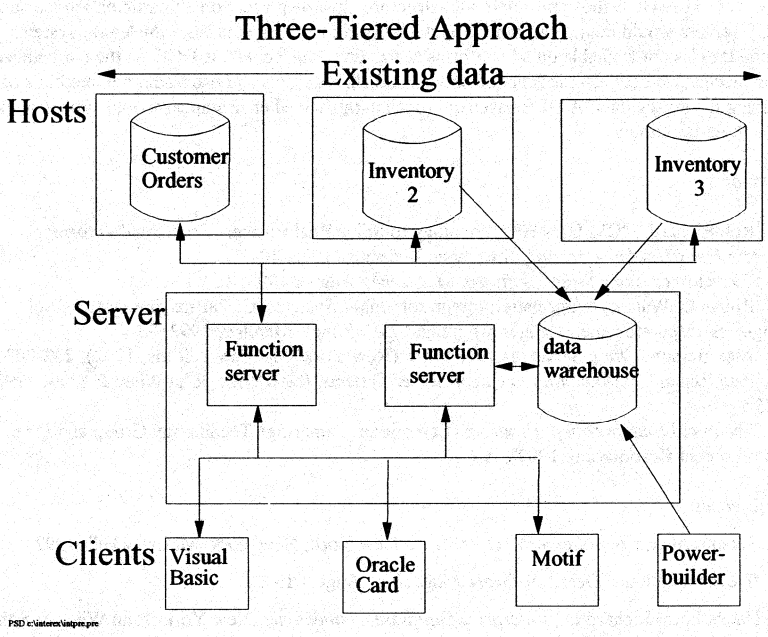


Figure 10

This solution uses a 3-tier approach based on RPCs. Functionality servers have been put in place to handle communications to the source systems. These provide real-time access to inventory data. In this case a customer order database is added to show that real-time data can be extracted from customer orders and inventory (perhaps to provide available to promise information for shipping) using a function server. In addition, another database is provided in the middle-tier to provide historical reporting. This database is queried using a functionality server that provides standard calculations which can be used by other clients. In this case, because of the functionality servers, a more varied client base can be supported. Lock-in at the database level is avoided. On-line access to the host systems is also provided. Overall, the approach provides more flexibility and also allows for the distribution of application logic. A two-tiered data access is shown with

the Powerbuilder client directly querying the data warehouse. This is a hybrid approach which combines the best of both two-tier and three-tier approaches.

DCE

In order to use DCE the appropriate infrastructure must be present on all participating machines. DCE servers would then be written on each platform to extract data from the legacy systems. Since DCE is not available on all platforms at this time, one could use DCE as the communication mechanism between the function servers and clients above and use traditional methods of accessing the legacy data. A DCE based approach in this mixed environment would appear similar to a three-tier model.

Endnotes

[1]Rob Richards, "CM Uses HP technology to Solve Real Business Problems!" *Information System News/Hewlett-Packard*, November-December 1992, 10-11.

[2]"Developing like a Deer," *Software Magazine*, August 1992, 22-24.

[3]Robyn C. Walker, "Information Integrator Unites Systems; CTPartners integrates Exel Logistics' disparate general ledger programs," *LAN Times*, March 9, 1992, 33.

[4]Alex Berson, *Client/Server Architectures*, (New York: McGraw-Hill Inc, 1992), 202-203.

[5]Paul Renaud, *Introduction to Client/Server Systems*, (NewYork: John Wiley & Sons, 1993), 175.

[6]*Technical Empowerment Program*,(Cambridge: Cambridge Technology Group and Open Environment Corporation, 1992), A-6.

References

Berson, Alex, *Client/Server Architectures*. New York, New York: McGraw Hill, 1992.

"Developing like a Deer," *Software Magazine*, August 1992.

Hackathorn, Richard D., *Enterprise Database Connectivity*. New York: John Wiley & Sons, 1993.

Renaud, Paul, *Introduction to Client/Server Systems*, NewYork: John Wiley & Sons, 1993.

Richards, Rob, "CM Uses HP technology to Solve Real Business Problems!" *Information System News/Hewlett-Packard*, November-December 1992.

Technical Empowerment Program,Cambridge: Cambridge Technology Group and Open Environment Corporation, 1992.

Walker, Robyn C., "Information Integrator Unites Systems; CTPartners integrates Exel Logistics' disparate general ledger programs," *LAN Times*, March 9, 1992.

Paper 6014

Using Application Response Times as a Metric for Service Level Agreements

Doug McBride
Hewlett-Packard Company
Performance Technology Center
Roseville, California

Background

Current day computer applications are, for the most part, interactive in nature. Whether it be simple text-oriented prompts and responses, or sophisticated high resolution GUIs (Graphical User Interfaces), the man-machine interface has moved from a batch orientation of the 50's, 60's and early 70's, to an interactive, on-line form of communication today that is almost universally taken for granted.

An important measure of how effectively the computer system is working for the user in an interactive application is **response time**. Response time is usually measured in one of two ways, referred to as **first response** and **to prompt**. **First response** is a measure of how long it takes the computer to get from a request for service (pressing a RETURN or ENTER key, a mouse click or drag, etc.) to the **first response**, or **output from the application** to the user. **To prompt** is the measure of how long it takes the computer to get from a request for service (again, the RETURN key, a mouse command, etc.) until it can **accept another command.**

The key interval in **first response** time is how long you have to wait until something appears that you can use to support the reason for the request. Not all information needs to be present before you start digesting what is appearing, so you can be working on what initially appears -- or at least be satisfied that something is happening -- while other information is being displayed. A measure of **first response** time is important for **information retrieval** types of applications, such as reading electronic mail, retrieving a parts explosion diagram, or getting inventory information on a particular part number.

On the other hand, in applications which require the computer be able to process a command or request and be ready to do another in a short period of time, **to prompt** becomes the important measure. You need to be able to process the maximum amount of work from the machine in the shortest possible time, so the machine must always be ready when you are. **To prompt** response time measures are important for **data input operations** where the idea is to be able to process the maximum input transactions per unit of time.

Ideally, all interactive applications would get instantaneous **first response** times so that no matter what you wanted to see, there was no or little perceivable delay in having that information appear. In addition, the computer would always be ready to accept the next

request for work, so you would never have to wait **to be prompted**. We all know from experience however, that this is rarely the case if the computer is being used close to its capacity limits (which is where a properly utilized piece of capital equipment should be running, right?).

Given the constraints placed on an application by available hardware capacity or software design, we also have to establish an upper limit for application response. If one goes beyond this limit, the application to loses its effectiveness as a business tool and at some point, fails to achieve the requirements it is supposed to meet in order to solve the business problem is was designed to solve. Even before the application meets the point of failure however, it will usually adversely affect productivity due to pacing, inconsistent service, and other problems that lead to poor human factor issues and reduced productivity.

Response Time in SLAs

Response times are an ideal measure for the establishment of Service Level Agreements (SLAs) for interactive applications. Since most mature data processing organizations are using, or are in the process of implementing SLAs, the response time metric is under scrutiny for its applicability in many applications.

The application owner should be able to specify what the maximum allowable response times (whether for **first response** or **to prompt**) for that application, as well as how many transactions of a particular type can fall outside of an acceptable range of response (percentiles). These response times can be derived by understanding the characteristics of the application, the problem it is intended to solve, the tolerances of the intended user audience to absolute response, acceptable variance, etc.. Based upon these specifications, service level objectives can be established by the data processing organization within the company, and the response time metrics tracked for compliance.

Sounds great to me! Anything this straightforward should be easy to specify and implement, right? These spiffy measurement tools everyone talks about should be able to give me the measured response times on a per-application basis, and I'm all set. Let's go do it! Satisfied users, here we come...

The Problem

By now you've probably figured out there's got to be a catch here somewhere, and of course, there is. As spiffy as our performance and resource management tools are, they don't do very well at response times. No provider of tools, regardless of platform (even IBM!), can easily provide response time information in a way that is totally meaningful to all applications. Why? Glad you asked. Let's explore the topic a little bit and see why.

What we really want to see in our response time measurements, whether to **first response** or **to prompt**, is data pertaining to a particular **transaction type**. A particular application may have one or two primary transaction types, or it may have several, all having distinct response characteristics and resource requirements. When we relate this to how our

measurement technology gathers the data to report response times, it becomes apparent where the problem lies.

How do we get the response time data the measurement and reporting tools give us? Here's a quick explanation:

First of all, the operating system must be instrumented (counters and trace facilities) to gather data pertaining to response times. In addition, there must be assumptions made by the engineers who write and insert this instrumentation regarding what devices will be used to support interactive applications to insure data is collected at the proper places.

Currently, those assumptions are that interactive applications are terminal based, so we capture information on how data moves to and from terminal devices. This may be on a character-by-character, line-by-line, or screen-by-screen (block mode) basis. Some assumptions are also made as to how the terminals are attached and communicate with the computer system.

If we measure the time from the **satisfaction of a read request** to the **next request for a write**, we have simple **first response** time information. If we measure the time from the **satisfaction of a read request** to the **next request for a read**, we've captured response **to prompt**. See Figure 1. below for an illustration of these response definitions.

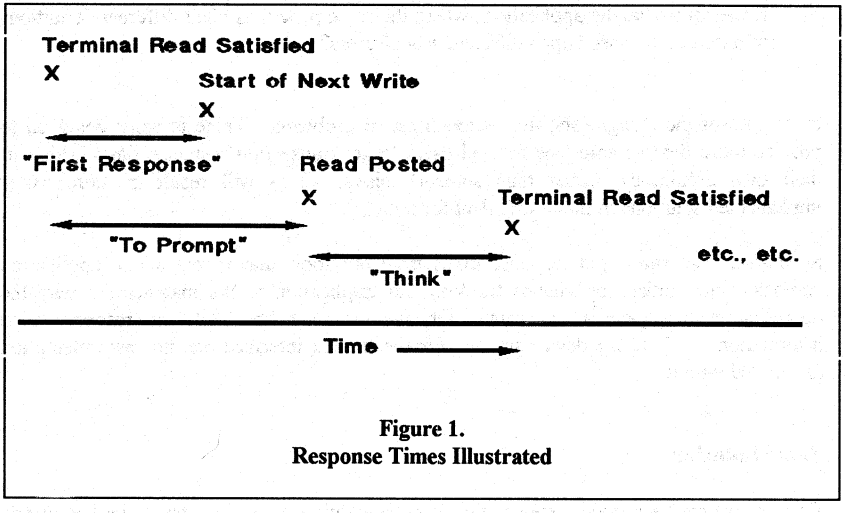


Figure 1.
Response Times Illustrated

The validity of these response times depends totally on the belief that a **transaction** is defined by the **simple assumptions** stated above. Although it's an easy thing to blame the hardware vendors or other providers of operating systems for not caring about us getting our valuable response time data, there are reasons for this weak link which I'll cover shortly.

I can see the wheels turning in your head at this point. What if our application's transactions:

1. Don't follow the simple definition of a transaction as posed above?

2. Isn't based on a terminal, and hence doesn't have response time instrumentation?

Obviously, our simple and straightforward response time situation just got more complicated.

On the other hand, consider some of the problems facing the engineers responsible for reporting response times out of the operating system:

√ How do we know what the logical definition of a transaction is from the user perspective?

√ How can we measure end-to-end response times when there is a network between the human interface and the computer?

√ How can we account for measuring response times on devices, the characteristics of which we may know little or nothing about?

√ How can we handle applications where the same programs elicit different definitions of transactions based upon different sets of users?

In defense of these engineers, this is tough set of problems. There is work going on to provide more flexible solutions for end users to get transaction response times which fit their own definitions, rather than someone else's. This will result in more easily implemented solutions in the future. But for now...

So anyway, we rarely get response times from our tools that reflect actual application response times unless by chance the way our application works matches the way the operating system gathers the data. Or, we may get absolutely no response time information at all if the device we use for the human interface has no instrumentation associated with it.

Some Solutions

Because we can't get good response times we shouldn't use response time as a key metric in our service level agreements right? No! There are other ways to get meaningful response times, as well as hope on the horizon for a better, more encompassing way to get this data. These include:

1. Being able to relate **our definition of a transaction** to the data we get from the operating system (post-process the data).

2. Telling the operating system where **transactions start** and **end** in our programs so it knows what we know.

3. Not asking the operating system to capture response times for us and instead, **do it ourselves.**

In the traditional world of mainframe performance analysis (spelled **I-B-M M-V-S**), the typical way to handle this problem is to post-process the data as mentioned in item 1. The instrumentation in their operating system or TP (Transaction Processing) monitors such as CICS, capture response times as described above. By understanding the relationship between the data captured by the operating system and how the transaction actually works, we can process the data to give us more "real" response times. Example:

Our application uses character-based prompts and replies to gather order entry data. We know it takes, on the average, 12 - 15 read-write transactions in the sense defined by the operating system. We'll call these a **physical** transaction. By summing the individual **physical** transactions that make up our **logical** transaction, we can get an idea of how long it took to process the entire order.

We can go back and study individual physical transactions if we need to hone in on a particular characteristic of the logical transaction. Or, we can just look at the average response time derived for our logical transaction as a measure of system response and provided service to application users. By having access to the data collected by the operating system on a per-application basis, this type of extrapolation is not difficult and gives a general idea of logical transaction response.

Another way of getting transaction response times relating to item 2 above, if we had a mechanism to tell the operating system or transaction processing monitor where our logical transaction starts and ends, the OS could compute the elapsed time for us and log it. This solves a big problem the operating system engineers have of knowing **exactly** what a transaction means to you. It also breaks free of the dependency to a known human interface device type, due to capturing **elapsed time**, rather than some **event** that deals with a **particular device** like a terminal.

Some intriguing opportunities come to mind regarding this mechanism, if the operating system engineers are clever (and of course, they all are, aren't they?). From a response time standpoint, we can easily live with elapsed time for our measurement, because that's a indicator of what level of service we are giving to our users. We can compare this elapsed time against our negotiated service level and if we're in tolerance, all is well with the world. But, if we're out of tolerance on a consistent basis, we have to fix the problem, or re-negotiate the service level agreement, right? What if the OS, in addition to giving us elapsed time, also gave us the associated **components of system response**?

What I mean by **components of system response** is identifying what the system was doing while processing the logical transaction and recording that activity. For instance, how much of the time was spent actually using a CPU in the system, versus queued for one (**service time** versus **time spent waiting**)? Same with other physical resources such as I/O or Memory. What about the use of logical resources which also have a service time and a potential waiting time associated with them (data base control blocks, network access queues, etc.)?

Total response time for a **logical transaction** is really just the **sum** of the time spent **receiving service** from some component of the system, and the time **waiting to access** that component. Where's a better place to find that information than in the OS instrumentation? If we can break out the components of a user-defined logical transaction, think of how quickly we could determine the cause of response time problems and potentially find a quick fix (or at least know where the problem was?).

To make our current situation more bearable, we know that work is ongoing in this area and should be available soon.

Finally, what I consider to be the best short-term solution at the time of this writing, is to take matters into your own hands and **JUST DO IT**! Using existing and reported Operating System response time data until newer, more state-of-the-art measurements are available is **NOT YOUR ONLY CHOICE**.

Assuming you have control over the source code for the application, it's not that difficult to insert your own instrumentation into application code. (This method, by the way, is the solution of choice for most shops who want to better manage their systems, and has been for years. Applications without this type of instrumentation are considered somewhat immature in mission critical environments.)

This local instrumentation is accomplished by locating the point in the application code where the application user starts the transaction through whatever mechanism is used. The wall time is then retrieved via an appropriate system call. The end of the logical transaction is then located in the same manner, and another call for the wall time is made. The difference between start and end wall time is the actual elapsed time for the transaction.

This information is written to a log file for later analysis by responsible resource managers. Since you may not want to always log these transaction response times for whatever reason, there usually is an application-global switch that is set through some configuration mechanism that enables the capturing of these response times. This is checked by some conditional statement in the code and either executed or ignored.

In this fashion, **YOU GET EXACTLY THE DATA YOU NEED** to support your service level management efforts.

Conclusion

Service Level Agreements are ideal vehicles for helping data processing management provide quality service to their end users. Response times are the ideal metric for measuring interactive applications from the perspective of the user within the SLA. Although other metrics might be of interest in maintaining the underlying service level objectives (component utilizations, device rates, etc.) for a particular SLA, the application manager will want to see how well you're doing in human, rather than computer terms.

As we've explored in this paper, response time measurements are not readily available that accurately reflect what application users think of as transaction response time. This is due to limitations in current operating system instrumentation based upon the rigid definition of a transaction. However, with a little imagination and a small amount of work, more meaningful response times can be derived either from exiting response time data, or by taking matters into your own hands and inserting your own instrumentation into your applications.

As resource managers demand more support from the operating system to get accurate response time data based upon their definition of a transaction, we should see better and more flexible ways of delimiting transactions in our applications. In turn, we can use the time saving logging and analysis features of most operating system performance and resource utilization monitors rather than doing it ourselves. In addition, with access to utilization and event data, operating system support of logical transactions has the potential of returning information on the components of response. This data can be quite valuable in better understanding how our applications use system resources, as well as what we might do to minimize that use and provide better service to our users.

I believe it's a given that if you're not already using SLAs, you'll be using them or something similar to them to help manage application performance via service level specification. SLAs are just too easy and too well defined not to take advantage of. To make your life easier in this regard, especially with interactive applications, raise your voice to your OS provider and make the suggestion they can make you a better manager (and hence, a better customer) by giving you the tools you need to effectively manage your system.

Good luck and keep the faith! We've come a long way in a short time, and suffered less than those who came before us (believe it or not). It WILL get better...

Paper No: 6015

Integrating EDI into your Business

Dr Trevor I.Richards
M.B.Foster Associates
9417 Great Hills Trail Suite 2038
Austin TX 78759
(512) 345 5376

I. Introduction

Electronic data interchange, or EDI, is a business decision which will change the way your company works. It will substantially alter the way that data flows into and out of your business applications. This paper explains the features commonly offered by third party EDI software and how these can be integrated with existing business applications to provide a true electronic interchange of data from your trading partner's application to your own, and vice versa.

The questions which will be posed during this discussion are -

- What is EDI?
- What is EDI software?
- What are the components of EDI software?
- How can EDI software be integrated with applications?
- What functions can applications provide to facilitate EDI integration?

II. What is EDI ?

In order to understand the need for some of the features offered by third party EDI software packages, it is first necessary to briefly discuss the current standards used to prepare EDI-formatted data, the methods employed to transport this data to and from trading partners, and the methods of acknowledging the integrity of the received data that are commonly used.

A. Definition

The standard definition of EDI is 'the automated, computer to computer exchange of structured business documents between an enterprise and it's trading partners'. The form of these documents is based upon the use of industry, national and international standards. However, this definition fails to emphasize the strategic impact of EDI upon a corporation, and is loose enough to enable many companies to participate in EDI without realizing it's full potential. A large proportion of today's EDI participants have not yet integrated the transmission and reception of EDI transactions with their business applications and are consequently failing to reap the considerable benefits associated with the elimination of duplicated data entry. Indeed, for some companies the savings gained from transmitting transactions to a trading partner, rather than sending them through the mail, have

been outweighed by the cost of re-keying outbound transactions, such as sales invoices, into EDI software packages which are used to prepare the EDI-formatted transactions. It is for this reason that I feel a better definition of EDI would be 'the automated, application to application exchange of structured business documents between an enterprise and it's trading partners'. This definition encompasses all of the major benefits to be gained from EDI.

B. Standards

Although the general perception is that EDI is a relatively new technology, the first recognized EDI transaction formats were developed by the Transportation Data Coordinating Committee, or TDCC, between 1968 and 1975. In EDI terminology, TDCC developed a "standard", which determines the methods employed to represent business documents in a structured form. The transaction formats which they defined were the first "transaction sets". By 1978, it was felt that the American National Standards Institute (ANSI) should support the development and maintenance of EDI transaction sets, and the ANSI Accredited Standards Committee X12 was formed. The first published transaction sets prepared according to the ANSI standard appeared in 1983 and have been revised several times to date. TDCC also encouraged the grocery and warehousing industries to develop their own transaction sets. By late 1982, the Uniform Communication Standard (UCS) was established and transaction sets were made available for the grocery industry. Around the same time the Warehouse Information Network Standard (WINS) was finalized and transaction sets were provided for use by the warehousing industry.

The period from 1978 to 1982 has been called the "enlightenment phase" where selected companies in selected industries began using selected EDI transactions. During the following years up to 1986, the "development phase" incorporated the expansion of the national standards to cover more industries. A similar expansion of national standards was experienced in the UK and Europe. The development of an international EDI standard began in 1985 and the first EDIFACT transaction set, or "message", was published in draft form in 1988.

Multiple standard formats for business transactions began to become a serious problem for those companies whose business crossed the artificial boundaries set up by the fore-mentioned regulatory bodies. As we will see later, industry bodies which were new to EDI began to develop guidelines for implementation of ANSI X12 transaction sets within their own industry rather than develop new EDI standard formats for their own needs.

There has also been a great desire in recent years to consolidate the national standards under one national EDI standard and accordingly the TDCC, UCS and WINS transaction sets are all currently in the process of redefinition according to the EDI syntax rules laid down by the ANSI Accredited Standards Committee X12.

The desire to adopt a single national standard is now beginning to encourage a

drive towards a single international EDI standard, EDIFACT. X12 transaction sets which employ the EDIFACT syntax are already in the process of definition, and the ANSI X12 membership recently voted to adopt EDIFACT as the single EDI syntax to be used for development of transaction sets after the release of Version 4 of the X12 American National Standard, which is expected in 1997. Most of the European national standards bodies have developed similar plans to those adopted by X12.

You may well be wondering how the history of standards development affects your own company and why it is relevant to a discussion of integration of EDI with business applications. The main reason, that you should be aware of the rich tapestry of EDI standards and transaction sets that are currently available, is that you may well have trading partners who require transaction sets from one or more of these EDI standards. Even if you are lucky enough to be a member of an industry where is single EDI standard like ANSI X12 is the only EDI syntax used for transaction sets, it is likely that your trading partners will have implemented different versions of the transaction sets. These facts can have a considerable impact on the features which you require of third party EDI software.

C. Industry-specific implementations

As I mentioned earlier, the trend for industries to recommend the method of implementation of EDI transaction sets to be used in their industry has replaced the desire to proliferate more EDI standards. The transaction sets defined by the EDI regulatory bodies such as ANSI X12 have now by necessity become extremely complex. For example, a single purchase order transaction set is required to service industries as diverse as the chemical and grocery industries. Both industries may use only 20-30% of the fields, or "data elements", defined for the transaction set and it is likely that these will overlap in very few places. Consequently, industry action groups such as the Automotive Industry Action Group (AIAG), the TeleCommunication Industry Forum (TCIF) and the Voluntary Interindustry Communications Standards (VICS) committee have recommended the data elements which should be used for each transaction set and what data they should contain. This has certainly been a contributory factor to the growth of EDI implementation over the past few years, as it has removed some of the uncertainty associated with establishing which data should be exchanged with your trading partner. However, if you trade with multiple industries, it adds another level of complexity and another variation to the format of a particular transaction set.

Consequently, you may need to establish from your trading partner for a particular transaction the following information :-

1. Which EDI standard is employed
2. Which industry-specific implementation guidelines were used
3. Which transaction set represents the transaction to be exchanged
4. Which version of the transaction set has been used

D. Communication Methods

In the early days of EDI, data was transported to a trading partner on magnetic tape, or a direct connection was established between the trading partners' computers. However, the variety of magnetic tape formats and communications protocols available made it impractical for a company to establish direct communications with every trading partner. Third party value added networks (VANS) provided the solution to this problem. They provide a store and forward mailbox service to allow transactions to be sent to and received from multiple trading partners who will probably have different computer systems, and who may be using different communications protocols. However, some of the major EDI trading partners, such as Chrysler and Wal-Mart, have found it practical to maintain their own communications network to establish direct connection to their trading partners, and for large volumes of transactions, particularly invoices which are not so time-critical, some trading partners still exchange data on magnetic tape. Third party EDI software should be able to accommodate any of these methods of data interchange.

E. Acknowledgement of data integrity

In order to ensure that the EDI transactions communicated to a trading partner were received and were acceptable in terms of the semantics and syntax of the EDI standard employed, the functional acknowledgement transaction was developed. This transaction informs the trading partner who transmitted the EDI transactions which individual transactions the receiving party actually received, and whether the format of the transactions complied with the semantics and syntax of the particular version of the EDI standard under which they were prepared. This obviously provides two more requirements for third party EDI software :-

1. Ability to create functional acknowledgements from inbound transactions
2. Ability to ensure that functional acknowledgements are received for all outbound transactions

There is also usually an agreed turnaround time for functional acknowledgements between trading partners, and it may well be that the third party EDI software package will be required to check whether each functional acknowledgement was received within this time limit.

A further level of acknowledgement which is starting to be employed, particularly in the automotive industry, is the application advice. This transaction takes the acknowledgement process one step further than the functional acknowledgement as it conveys information on either the acceptance or rejection of a transaction by the receiving application, and the reasons for rejection if this occurs. In the long term, this method of informing the sending party of transactions rejected from the receiving application will become commonplace.

III. What is EDI Software ?

Now that we have looked at a brief overview of the development of EDI standards, methods of communication and acknowledgement of data integrity, we can consider the reasons for employing third party EDI software to manage the interface with trading partners.

However, before we are able to discuss the benefits of using EDI software, we need to look at the total process from the originating application to the target application. Trading partner A originates a transaction from his application. It is transformed into an EDI-formatted transaction and sent to trading partner B's mailbox on a value-added network using an EDI software package. Some time in the future, trading partner B retrieves the data from his mailbox and transforms it into a format which his application can read. Once again an EDI software package is used to execute the communications and data transformation. Now let us look at the benefits of using a third party EDI software package.

A. Bridge to EDI Standards

Third party EDI software provides a bridge from the internal business systems environment, where data is held in proprietary formats, to the outside world, where transactions are now being electronically communicated as EDI transactions formatted according to one of the EDI standards.

B. Insulates applications from EDI Standards

As was mentioned earlier, the EDI standards have developed over the past two decades and they continue to be refined. Each transaction set defined under a particular standard may have several versions which have been implemented and which are currently in use. Third party EDI software should provide the user with the ability to define neutral or generic file formats for inbound and outbound transactions which can be transformed into any of the required versions of the EDI transaction sets. If this can be achieved, it enables the user to insulate internal business applications from the ever-changing external world of EDI standards.

C. Meets needs of multiple trading partners

A further level of sophistication which needs to be made available is the ability to satisfy the needs of multiple trading partners from the same internal inbound or outbound generic file format. The user should be able to specify to the EDI software package which version of a particular EDI transaction set should be sent to or received from a particular trading partner. The user should then also be able to specify that this trading partner expects to send or receive transaction sets prepared according to a particular set of industry-specific implementation guidelines, or that this trading partner has a unique set of requirements for the implementation of this transaction set.

D. Converts variable input into predictable fixed formats

Bespoke and third party business applications generally manage files or databases consisting of fixed length records comprised of fixed length fields. EDI transactions are constructed from variable length records, or "segments", which are constructed from variable length fields, or "data elements". EDI transactions can be further compressed by concatenating and splitting segments to fit into fixed length records for data transmission. EDI software provides the tools to unwrap the compressed form of an EDI transaction and convert the variable length input data to a more familiar fixed length format.

E. Integrates EDI data with business applications

The overriding objective of third party EDI software should be to provide the user with the ability to manage the transformation of application files to EDI transactions, and vice versa, and to control the communication of these transactions to and from the user's trading partners. This should be achieved with the minimum of disruption to the business applications involved.

IV. What are the components of EDI software ?

Now that we have discussed what benefits third party EDI software should provide, we should consider what component parts may be provided by the vendor. The following discussion is based on a survey of the features offered by third party EDI software across multiple hardware platforms and represents the functionality options which you should expect to find.

A. Communications software management

The extent of integration of communications software management within an EDI software package can vary from a fully integrated communications environment to none at all. The importance that you place on the presence or absence of this functionality is likely to depend on the frequency of communication with your trading partner or VAN. In the automotive industry, for example, EDI transactions can be transmitted to and received from the same trading partner many times each day. In this environment communications software management is desirable.

1. Communications software

Software may be provided by the vendor to drive various different communication protocols. However, the desirability of this feature will probably be determined by whether or not you already own communications software to achieve the desired communications protocols, and whether the third party EDI software package is capable of integrating with this software.

2. Job control

The features offered for control of the transformation process can be segregated into three areas :-

a. communication session
b. pre-communications activities
c. post-communications activities
d. error handling

Control of the communications session generally involves selection of the communication method, together with the means to construct and execute command sequences to control the communications software and the dialogue with the value-added network.

The pre-communications activities will generally be relevant to outbound transactions, such as the transmission of sales invoices, and may execute the processes associated with the extraction of data from the application system and transformation of that data to the required EDI transaction format.

The post-communications activities will generally be relevant to inbound transactions, such as the receipt of sales orders, and may execute the processes associated with transforming the received EDI data to the required application file format and uploading that file into the application.

The error handling component of job control should allow the user to define what actions should be automatically taken if errors occur in the activities performed before, during and after the communications session.

3. Job scheduling

The job scheduling component offers the capability to schedule jobs to occur at pre-defined intervals, to review the job schedule, and control jobs currently executing. It may also have the ability to reschedule jobs which have failed due to an inability to communicate with the trading partner or VAN.

B. Data Transformation

The data transformation capabilities of an EDI software package are the heart of the product. They can also be the part of the product where the terminology employed by software vendors is most confusing. For the purpose of this discussion I will try to explain the terms I have used for the data transformation functions.

1. Translation:

- transforming the variable formats associated with EDI transactions into pre-defined fixed length records, and vice-versa.

2. Remapping:
- transforming the pre-defined output from the translation process into a user-defined file format which is capable of being interfaced with the user's application, and vice-versa.

3. Mapping:
- transforming the variable format associated with EDI transactions directly into a user-defined application file format, and vice-versa.

All products offer at least the translation of EDI transactions to pre-defined fixed length records. In some products remapping of these pre-defined records to user-defined file formats is also offered, whilst in others the translation and remapping processes are combined to allow direct mapping from the EDI transaction to a user-defined file format. The major benefit that remapping or mapping to a user-defined file format brings is flexibility.
The potential user of an EDI software package needs to clearly understand what data transformation capabilities each software vendor offers, as there are no definitions of the above terms which are universally accepted. If transformation to and from user-defined file formats is an important issue, for example, you need to clearly establish that this facility is available.

C. Transaction management
In addition to communications and data transformation, an EDI software package may offer facilities to manage the diversity of your trading partners' requirements, together with facilities to manage the flow of transactions to and from your trading partner.

1. Trading Partner Profiles
Trading partner profiles which can be maintained down to the level of individual transactions may be provided. The parameters held in these profiles may include which VAN is to be used, which standard employed and which industry-specific implementation is to be used for this trading partner. At the transaction level parameters may include which version of the transaction set is to be used and whether or not a functional acknowledgement is required.

2. Functional Acknowledgement Tracking
For outbound transactions the EDI software package may provide the ability to record functional acknowledgements received from your trading partner and determine those outbound transactions which are still awaiting acknowledgement. This may also be further refined to add the ability to determine whether those functional acknowledgements which have been received were received within the agreed time limit.

3. Security

The EDI software package may offer various levels of access to database controlling EDI activities, such as who can initiate jobs, who can create jobs, etc.

4. Control/Audit Reports

Reports which detail the flow of transactions through the complete inbound and outbound processes may be provided, and may include transaction count and value hash totals to aid reconciliation.

5. Archiving

The software package could also provide a mechanism for archiving the data files associated with the complete inbound and outbound processes to a backup medium such as magnetic tape before deletion from the system.

V. How can EDI software be integrated with applications ?

Having looked at the functions which are offered by third party EDI software packages, we can detail the processes through which both inbound and outbound transactions pass. I will concentrate specifically on the processes conducted by third party EDI software which facilitate it's integration with both the value-added network and the application system.

A. Inbound transactions

A typical example of an inbound transaction would be a purchase order from a customer which needs to update the sales order processing system as the target application. This transaction would normally be entered through on-line data entry, and it is useful to remember this fact during the following discussion.
An inbound document such as a sales order would pass through the following processes :-

- Receive trading partner's data from the trading partner/VAN
- Map EDI transactions to an application interface file
- Upload application interface file to application via batch interface

1. Receive data from trading partner/VAN

- A communication session using an appropriate protocol is established with the trading partner or VAN, and data is received
- The received data is analyzed and non-EDI data, such as the mail box listings from the network, is segregated from the EDI data and printed
- The packed EDI transaction data is unpacked from a block of fixed length records into it's constituent variable length segments

2. Map EDI transactions to application interface file

- The EDI data is then checked to ensure that the transactions received comply with the semantics and syntax of the standard under which they were created
- If errors are found, actions associated with the occurrence of the error messages should be initiated at this point. These could vary in severity from notifying an appropriate user of a non-critical warning to the termination of further processing
- The compliance checking that has now been conducted allows the generation of functional acknowledgements for these transactions
- The fields of the records in the application interface file can now be populated according to the rules laid down for this trading partner.

  Mapping is required because there is not usually a direct relationship between all of the fields required for the application interface file and the data elements in the EDI transaction.
  These fields must be generated by the following methods :-

  a. Map an inbound data element directly to an output field
  b. Look up the data element in a cross-reference table and substitute the alternate value in the output field
  c. Carry out string manipulation or arithmetic on one or more data elements to produce the output field
  d. Calculate a variable such as today's date and place in output field
  e. Enter a default value in the output field

  The mapping process parallels the traditional process of manual data entry, where human beings perform the cross-referencing using additional files, calculate line extensions, or use common sense to determine reasonable default values.

3. Upload to application via batch interface

- Validation comparable to the equivalent on-line transaction is performed on the data in the application interface file
- The application is updated with valid transactions and invalid transactions are placed in an error suspense file
- Any erroneous transactions held in the suspense file are rectified by either correcting the data in the error suspense file, or correcting any errors in the application master files
- Once the errors are rectified, the contents of the error suspense file are resubmitted to the batch interface
- This process is continued until all the transactions are loaded

B. Outbound transactions

A typical example of an outbound transaction would be a sales invoice to be sent to a customer which originates from the sales order processing system. An invoice would normally be printed from the application system and mailed to the customer.

An outbound document such as a sales invoice would pass through the following processes :-

- Extract data from application databases to application interface file
- Map application interface file to EDI transactions
- Transmit data to trading partner/VAN

1. Extract data from application to application interface file

- Only those transactions which are associated with trading partners who wish to receive these outbound transactions via EDI are selected
- For these transactions, the application interface file is populated with data from the application databases
- Checks are made to ensure that no critical data in the application interface file has been omitted. Any transactions which lack critical data items are removed from the application interface file
- The application databases are updated to indicate these transactions are to be transmitted via EDI

2. Map application interface file to EDI transactions

- The data elements of the EDI transaction set can now be populated according to the rules laid down for this trading partner.
  Mapping is once again required because there is not usually a direct relationship between all of the data elements required for the EDI transaction set and the fields of the application interface file. These fields must be generated by the following methods :-

  a. Map application file field directly to output data element
  b. Look up field in cross-reference table and substitute alternate value in the output data element
  c. Carry out string manipulation or arithmetic on one or more fields to produce output data element
  d. Calculate a variable such as today's date and place in the output data element
  e. Enter a default value in the output data element

- The control segments of the EDI transaction set will then be generated, and unique control numbers associated with the control

points in the EDI transaction set will be generated according to the rules laid down in the trading partner profiles

- A record is then created to indicate that a functional acknowledgement is required for this transaction

3. Transmit data to trading partner/VAN

- Before the data can be transmitted, it must first be segregated into groups of transactions to be sent to each trading partner or VAN according to trading partner profiles
- The variable length segments produced by the mapping process are then packed into fixed length blocks of data for transmission to the trading partner or VAN
- The compressed data is then sent to the appropriate trading partner or VAN using a common communications protocol

VI. What functions can applications provide to facilitate EDI integration ?

The preceding discussion of the processes involved in integrating third party EDI software with internal business applications leads to the following conclusions about the functions that can be provided by these applications.

A. Inbound transactions

Batch interface should have :-

1. ability to load transactions from application interface file
2. batch validation which duplicates or extends current on-line validation
3. error suspense file creation capability
4. error correction and resubmission capability

B. Outbound transactions

Data extraction should provide :-

1. ability to segregate transactions according to communications medium (ie is this purchase order to be printed and mailed to trading partner, or sent via EDI, or both)
2. application update to indicate that transaction was sent and that it was sent via this communication method

VII. Summary

Third party EDI software should not just include translation. It should ideally manage your communications with VANs, transform EDI transactions to application interface files and vice versa, and control an environment which may include many trading partners using many different implementations of EDI transactions sets. EDI software should provide the capability to integrate EDI with your business applications with the minimum of disruption. However, business

applications need to provide some functionality to allow EDI integration, and the simple rules to be learnt from this brief discussion on integration are :-

A. for inbound transactions, which replace data normally entered through on-line data entry, the business application must provide the necessary functions to load transactions through a batch interface and ensure the integrity of this data

B. for outbound transactions, which replace data normally sent to the trading partner on a hard copy document, the business application should provide a means to identify that a particular transaction should be sent via EDI, and should update the transaction record to indicate that is has been sent via EDI

PAPER # 6016

Exploring HP-UX - An Investigation into its Performance Envelope Especially for MPE/iX Fans

by Robert A. Lund
34130 Parkwoods Dr., N.E.
Albany, OR 97321
(503) 327-3800

In this paper I would like to accomplish a couple of things. First, for you to become familiar with some of the more important HP-UX and MPE/iX performance terminology... how they compare and how they differ. This will be especially helpful if you have HP 3000 experience and how you are dealing with, or soon will deal with an HP 9000 system. Next, I will present a simple synthetic case study which involves pushing an HP 9000 series 807 to the corners of its performance "envelope". Here, I will present some Key Performance Indicators (KPIs). I particularly enjoy stressing systems with synthetic loads and making observations, so bear with me if I come off as having too much fun! We'll focus specifically on the relationship between CPU and memory shortage on system performance; how they affect response time and transaction throughput.

A brief encouraging word. Most of the issues on an HP-UX system have parallels on an MPE system. A rose is a rose... Apart from feeling a little stupid with some of the commands and terminology, probably the worst thing that will happen is that you will have to get used to typing in lower case!

A SHORT REFRESHER COURSE IN PERFORMANCE BASICS

Consider the following,

"The subject of system performance on any computer system can be very technical. It involves the interaction of an amazing amount of convoluted software and hardware mechanisms. This interactivity is supposed to help you produce information that will "oil" the wheels of your company. The entire process must be manageable and it must be cost-effective. " [1]

This is where you come in! You need to be sure that your system keeps satisfying your customers (users, management, etc.). You accomplish this by agreeing upon, measuring, and managing service levels. Service has four aspects: Timeliness, Accuracy, Cost, and Reliability. We're focusing on the timeliness angle. This means acceptable user response times and batch processing (in HP-UX batch processing is sometimes also referred to as "Chron jobs") turnaround time. Typically, the best way to provide good service is as follows:

1. Measure the current performance (objective monitoring and subjective user input) - This may involve historical trend analysis also.

2. Tune and tweak the system (this may involve upgrading, file maintenance, load balancing, etc.).

3. Re-Measure performance and compare with user expectations.

This entire process is somewhat iterative and may represent a long or short cycle depending how volatile performance is at your shop.

Covering the "Bases"

To effectively and consistently deliver good service to your customers, you must cover all the performance bases. That is, you have to deal with four areas:

1. Crisis Mode: What plan does your staff have in place when the system is "hung" or background processing is "dead in the water"?

2. Casual Mode: What tools/methodologies do you have to allow your staff to casually monitor the system in a meaningful fashion?

A short note here. In case you have not already noticed, UNIX is quite cryptic if you are used to MPE/iX. It actually is much better and more flexible in many ways, but nevertheless will require you to get used to cryptology...er uh, I think this is the science of taking meaningful English commands and shortening them and putting them into lower case so that everyone thinks you are really smart when you type them in! Some of the system commands have performance data, but they are anything but easy to understand. sar, ps, iostat, vmstat, etc are all examples. I am seeing many more people desiring useful tools to help present and interpret this data in such a fashion that it will be meaningful for those whom do not have a Ph.D. in cryptology!

3. Capacity Planning - What will happen to user response times, I/O throughput, CPU utilization, etc. when you add more applications, users, etc.?

4. Problem Solving - How do you profile new applications as to their performance appetite? What tools do your programmers have to help them solve application performance problems?

These are the main areas where you need to focus your "performance attention".

Primary System Resources

Although there are usually many different resources necessary to complete transactions on any computer system, the three main ones are the CPU, primary storage (main memory), and secondary storage (disk devices). Some amount of each of these resources can be thought of as ingredients needed to perform a unit of useful work (a transaction).

Bottlenecks

When any necessary resource becomes in short supply, that resource becomes a choke-point (referred to as a bottleneck). Often, processes (programs) then have to wait for that resource to become available. I have had some fun in recent years studying queueing theory, which is the science of bottlenecks... An interesting phenomena that takes place on busy systems is what is referred to as...

The Knee in the Performance Curve

You have probably heard of this term from time to time. Its basic meaning has to do with a small increase in load resulting in a surprisingly large increase in response time or job turnaround time. With that definition in mind, take a look at Figures 1-4.

Figure 1 - HP-UX (HP 9000 807) Synthetic Test; Memory Load=250

These figures are the result of some synthetic tests performed on an HP 9000 807. A load was configured, run on the system (a 10 minute run for each number of user processes), performance metrics logged, and then the next series was commenced. This resulted in some interesting data.

Figure 1 represents a light memory load. Each simulated user was configured to consume a specified amount of CPU, memory, I/O accesses, etc. Between figures 1 through 4 the only thing that varied was the amount of memory each user used. Notice the gentle response time "knee" in figure 1 at the 9 user level. What this graph is representing is this: as 3 more users are added (starting with a single user), the amount of response time they experience rises linearly at a comfortable level (from .6 to .9 seconds) up to 9 users. Notice the proverbial knee. At this point adding three more users to this workload on this HP-UX system, subsequent response times rise higher than the preceeding trend. Even so, 1.4 seconds is probably acceptable. Notice that the number of transactions completed continues to rise after this point.

(ed note: figures 2-4 could go anywhere here)

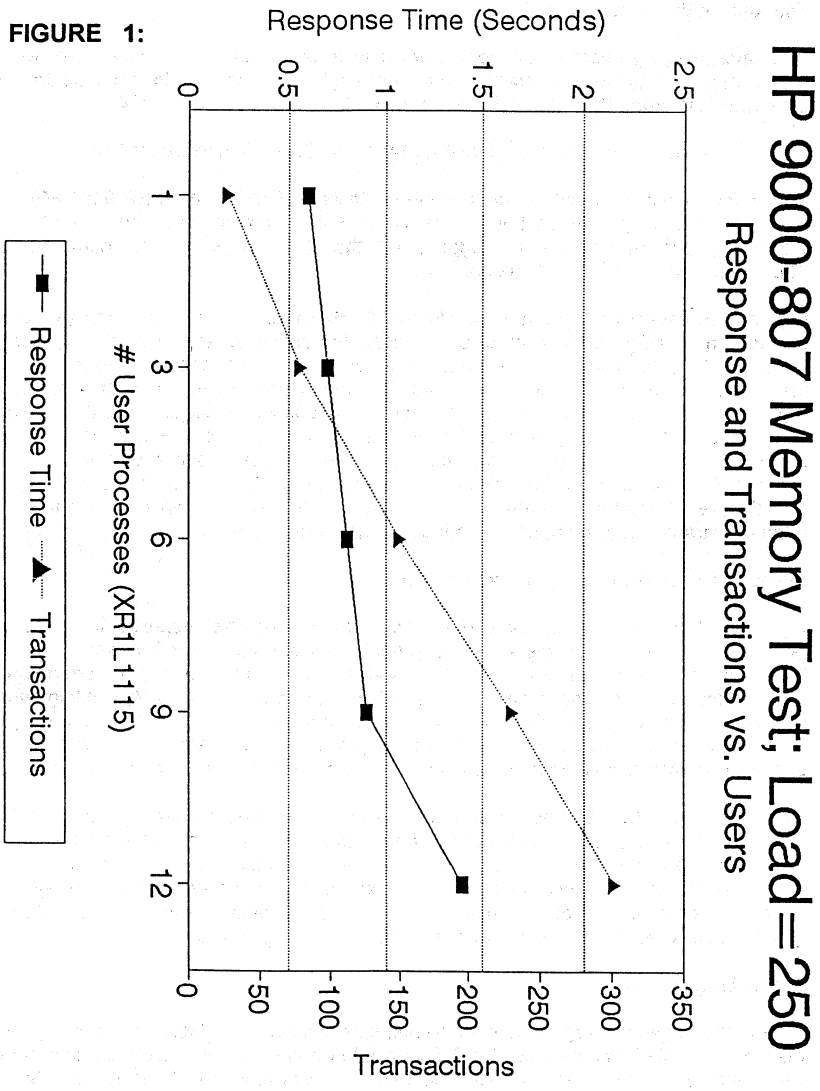
Figures 2-4 become increasingly dramatic. Notice that the response knee points become very steep. Also, the knees keep getting pushed back to fewer users. Take a look at the transaction knees in figures 3 and 4! After some point in a performance curve, throughput decreases due to a dimishing returns effect. Namely, adding more users after resource saturation causes less transaction completions in the same amount of run time. And in figure 4 transaction throughput becomes flatter. This means that even with an increase in the number users, more transactions are not accomplished.

One of the scary things about these knees, is that many systems are "flirting" with the knee in their performance curve and don't even know it! (ed: this next sentence would be a good one to put in a cool highlighted box). A steep knee in the curve means that it is possible to add just a few more users to an application and simply cause your system to go down on its knees (pun intended!). Figure 3 is a dramatic example of this. At the 12 user level, response times sky-rocket and transactions plummet.

The CPU Resource

After a short diversion regarding bottlenecks, let's continue our discussion about system resources; the CPU will be first. On RISC-based systems (9xx and 8xx in particular) the CPU tends to be the most common bottleneck. It was one of HP's design policies to shift, as much as possible, all bottlenecks to the CPU. Of course, there are many exceptions to this, but, generally speaking, the CPU tends to be the culprit when performance is not up to par.

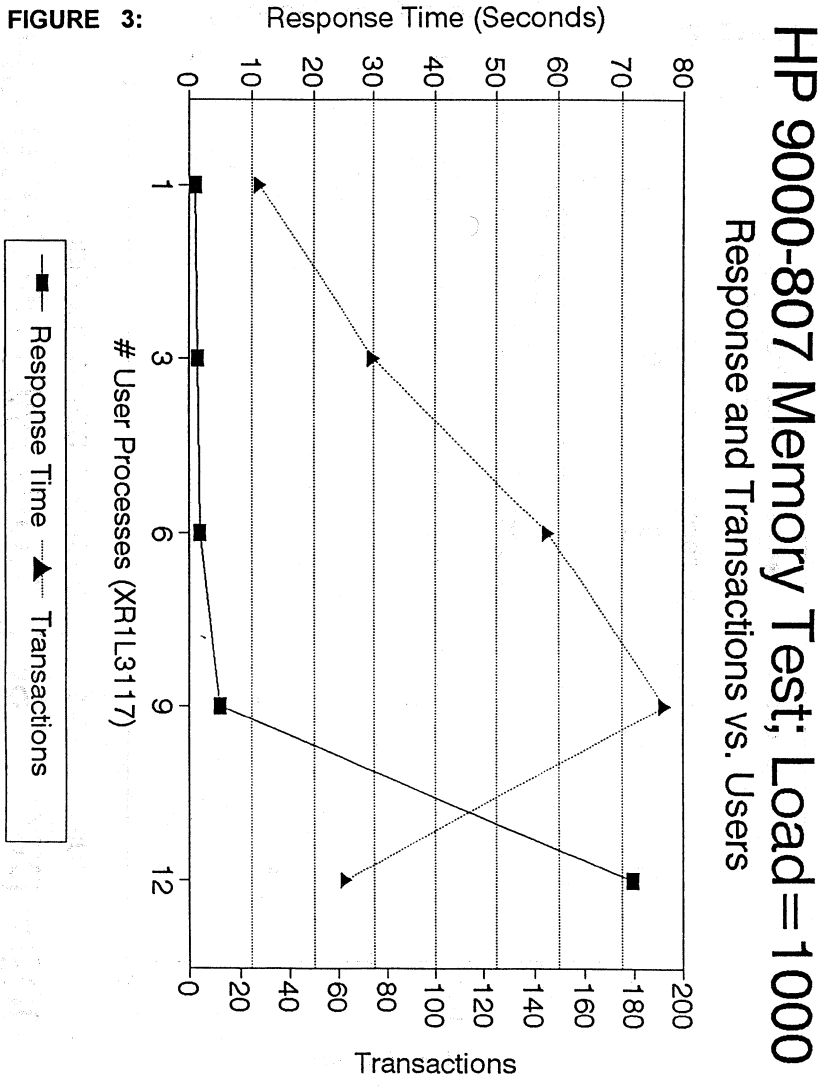
FIGURE 1:


HP 9000-807 Memory Test; Load=250
Response and Transactions vs. Users
Response Time (Seconds)
0
0.5
1
1.5
2
2.5
Transactions
0
50
100
150
200
250
300
350
User Processes (XR1L1115)
1
3
6
9
12
Response Time
Transactions

HP 9000-807 Memory Test; Load=500
Response and Transacations vs. Users
Response Time (Seconds)
Transactions
User Processes (XR1L2116)
Response Time
Transactions
0
0.5
1
1.5
2
2.5
3
3.5
4
50
100
150
200
250
300
1
3
6
9
12

FIGURE 2:

FIGURE 3:


HP 9000-807 Memory Test; Load=1000
Response and Transactions vs. Users
Response Time (Seconds)
Transactions
User Processes (XR1L3117)
Response Time
Transactions

HP 9000-807 Memory Test; Load=2000
Response and Transactions vs. Users
Response Time (Seconds)
Transactions
User Processes (XR1L4118)
Response Time
Transactions
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
80
90
100
1
3
6
9
12

FIGURE 4:

Table 1 includes some CPU "measuring sticks" for both MPE/iX and HP-UX. Use these to help you determine bottleneck conditions. I've tried to include some rough rules of thumb to assist you in identifying "red zone" conditions.

MPE/iX Indicator	HP-UX Indicator	Red Zone	Comments
AQ-EQ Busy		*	Note 1
	User Busy	*	Non-Niced processes
	Real Busy	*	Very high priority linear processes
	Nice Busy	*	Processes that have a low priority (Note 2)
	System	*	Kernel processing time (includes memory)
Memory Mgmt		>12%	Great memory shortage indicator
Dispatcher	Context Switch	#	>15% when added with ICS or Interrupts
ICS/Overhead	Interrupts	#	See above
Pause	Pause	>15%	Disk I/O bottleneck indicator
Idle	Idle	?	Lots here is good! CPU in the bank!
CPU Queue	Run Queue	>8-15	No. of processes waiting to get CPU time (somewhat system size dependant)

Note 1: On MPE/iX systems process busy time is broken out by priority queue. For HP-UX you will use Real, User, Nice, and System. One main rule of thumb for CPU saturation is if your high priority processing consistently exceeds 85% or so, this is a sign of CPU shortage. Response times and throughput could suffer as the system gets busier.

Note 2: Nice processing is basically programs that have had special priority degradation. This will affect programs that tend to be CPU hogs. If a positive NICE value is applied to a program, it will be effectively lowered in priority. This keeps such programs (typically CPU intensive) from taking over the system.

Table 1 - MPE/iX and HP-UX CPU Performance Metrics

A CPU-INTENSIVE SYNTHETIC CASE STUDY

In this study we will look at a series of increasingly heavier CPU-consuming processes. This experiment started out with one user process having a set appetite of CPU, I/O and memory. Simulated users were added 3 at a time for subsequent tests. Each time the amount of CPU consumed by each user was increased. Thus, for figure 5 we studied 1 through 18 simulated user processes. Each of these pseudo-users had a specified think time, CPU usage, disk access, and memory consumption.

Figure 5 - CPU-Intensive Synthetic Study - Detail CPU Utilization

Notice in Figure 5 that somewhere around the 9 user mark the CPU is consistently above the 85% range. Even though this processing is considered NICE processing (running at a low priority), since these programs were the only ones running on the system, they are considered high priority for this experiment's sake.

Figure 6 reveals what happens on this 807 system after the system consistently hits the 100% CPU busy mark. It's the same problem you face at McDonalds at 12:00 noon. The cooks and service folks are moving at capacity...they simply cannot work any faster,

HP 9000-807 CPU Test; Load=2000
Detail CPU Utilization (1-18 Users)
CPU Utilization
100
90
80
70
60
50
40
30
20
10
0
1 User
3 Users
6 Users
9 Users
12 Users
15 Users
18 Users
14:05
14:11
14:16
14:21
14:26
14:32
14:37
14:42
14:48
14:53
14:59
15:04
15:11
Time (XC3L5064)
User Process
Nice Process
Overhead
System Process
Pause for Disk

FIGURE 5:

so customers wait in line longer. Peak demand times are the bitter/sweet experience of probably every fast-food business! The same is true for your customers; when the system gets saturated, user requests for CPU time simply queue-up (as they say in the UK!). This is illustrated in Figure 6. Notice how the CPU Run Queue simply grows and grows...

Figure 6 - CPU Intensive Synthetic Study - CPU Busy and Run Queue vs. Growing Users

Figure 7 shows the felt effects of a growing CPU problem. At about the 9th user the total number of transactions completed begins to drop off. Why is this? A hint can be seen in figure 5. Note in that figure that a bit of "System" CPU time begins to raise its ugly head. What this means in plain terms is that the system is simply getting overwhelmed (thrashing) with all of the housekeeping associated with an absurdly busy system. This not only robs valuable CPU time, but indicates that the load is simply too large for the system to handle and still produce acceptable response times. Notice how response times grow from around 10 to 150 seconds per transaction!

Figure 7 - CPU Intensive Synthetic Study - Response and Transactions vs. Growing Users

Figure 8 is included to help the MPE or novice HP-UX user become aquainted with various kernel activities.

Figure 8 - CPU Intensive Synthetic Study - Misc System Activities

Some definition will help with regard to the system activities shown in Figure 8. A context switch occurs when the kernel dispatching mechanism gives attention to a different process. This may occur many times per second. On MPE systems this activity is analogous to a process launch. A fork is similar to a process creation event on MPE. A trap occurs when a process performs some interruptive activity like a system call, etc. An interrupt is just like that on MPE; it is a hardware generated CPU interruption (terminal, disk drive, etc.). System calls are like MPE system library (SL, XL, NL) calls.

It can be seen from the foregoing CPU observations that a few metrics will help you monitor your system's performance more effectively. Try to keep your eye on the three stages of the CPU: Busy, Paused for I/O, and Idle. Some CPU statistics can be derived from system commands such as sar, vmstat, and iostat, but, staying with the cryptic UNIX tradition, they are often difficult to read and interpret. More importantly, they certainly do not provide managers with the kind of data that is needed to make strategic performance decisions. There are a few third party tools such as SOS/9000 Performance Advisor that will provide much more meaningful performance data.

A MEMORY-INTENSIVE CASE STUDY

Now let's turn our attention to the subject of memory. As you probably know, adequate memory is vital in order to insure good performance on nearly any computer system. When in doubt, over-buy a bit on memory if you have any money left in the budget. The following figures represent some memory statistics derived from a synthetic study on a

HP 9000-807 CPU Test; Load=2000
CPU Busy and Run Queue vs. Users
CPU Utilization
CPU Queue Length
100
90
80
70
60
50
40
30
20
10
0
20
18
16
14
12
10
8
6
4
2
0
14:05
14:11
14:16
14:21
14:26
14:32
14:37
14:42
14:48
14:53
14:59
15:04
15:11
Time (XC2L5059)
Total CPU Busy
CPU Queue Length

FIGURE 6:

FIGURE 7:

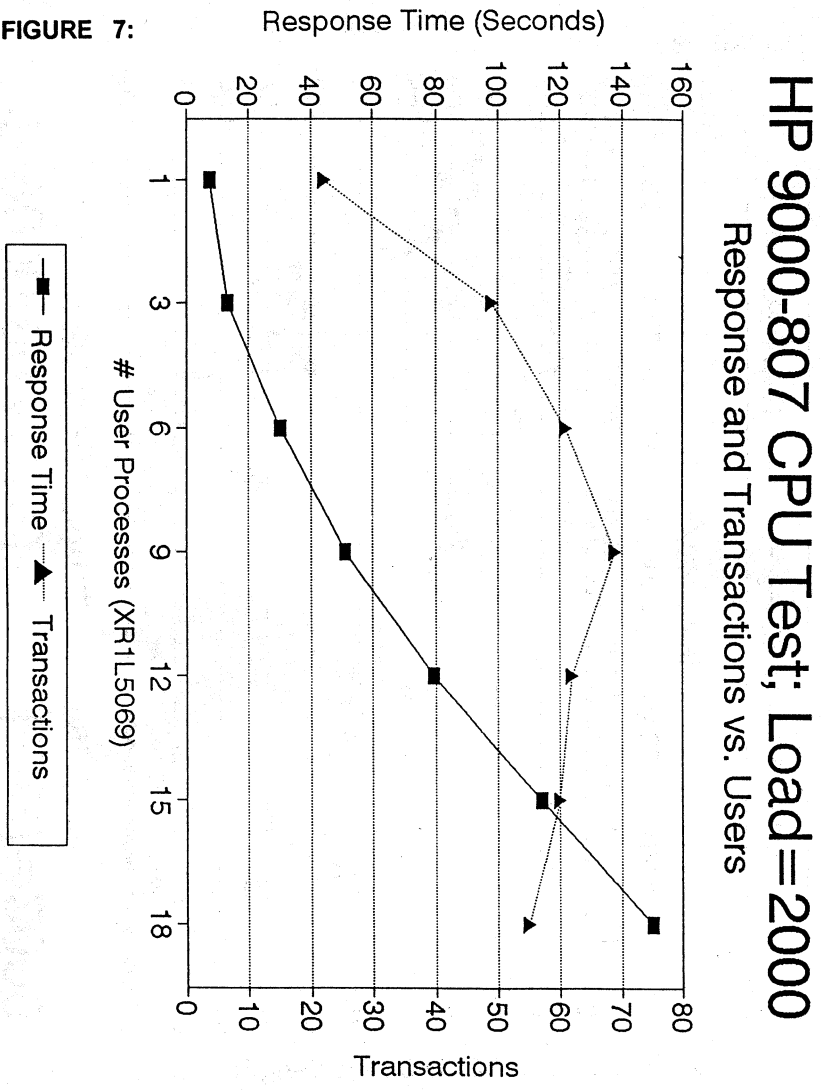
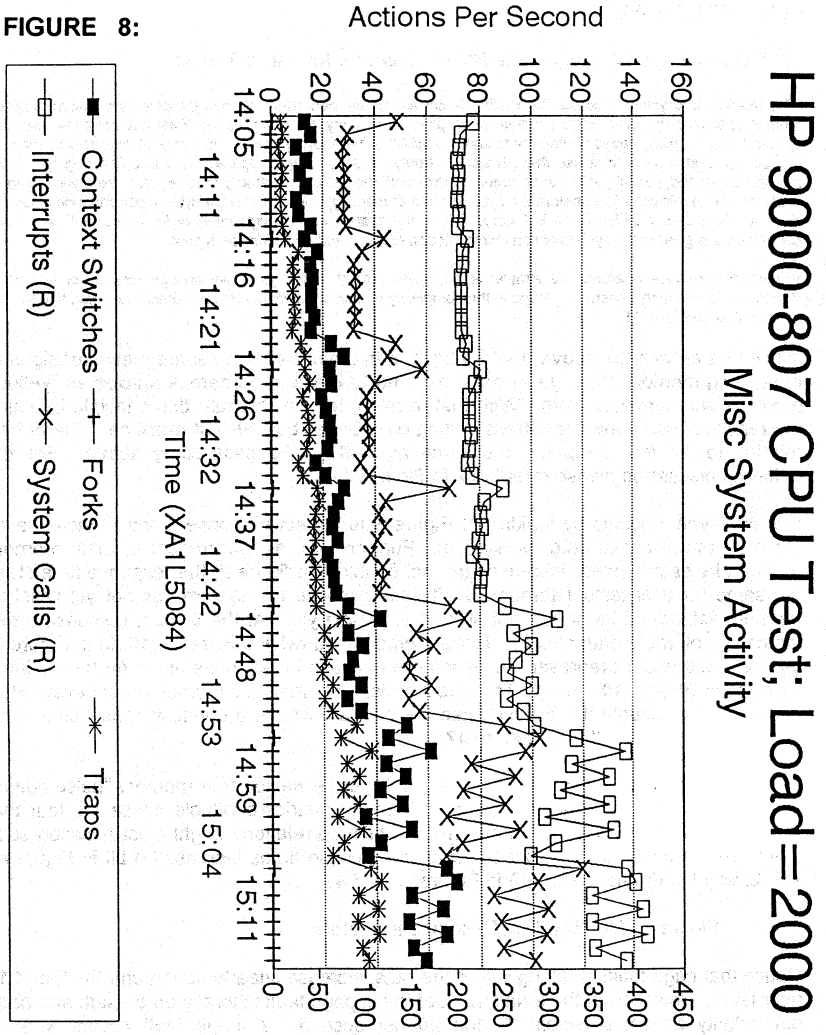

HP 9000-807 CPU Test; Load=2000
Response and Transactions vs. Users
Response Time (Seconds)
Transactions
User Processes (XR1L5069)
Response Time
Transactions
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
60
70
80
1
3
6
9
12
15
18

FIGURE 8:


HP 9000-807 CPU Test; Load=2000
Misc System Activity
Actions Per Second
Ints & Sys Calls Per Second
Time (XA1L5084)
Context Switches
Forks
Traps
Interrupts (R)
System Calls (R)
0
20
40
60
80
100
120
140
160
0
50
100
150
200
250
300
350
400
450
14:05
14:11
14:16
14:21
14:26
14:32
14:37
14:42
14:48
14:53
14:59
15:04
15:11

series 807 HP 9000.

First, consider the following quote from System Performance Tuning,

"The memory subsystem becomes the limiting factor for system performance when the programs that are actively running (including the UNIX kernel) require more physical memory than is available. Past this point the operating system begins copying pages of memory to and from disk. After copying one or more pages from physical memory to disk, the system gets to re-use that physical memory for some other purpose. This is called paging. Once paging has started, overall performance drops sharply until the system's memory requirements are again within its capacity. UP to this point (i..e., before the system is out of memory), memory is not a system performance issue. In other words, memory performance is binary. Either the system has enough memory for its current needs and performance is good or the system doesn't have enough memory and performance is poor.

Memory performance problems are simple: at any given instant, either you have enough memory or you don't. However, this summary doesn't do justice to the complexity of memory management nor does it help you to deal with problems as they arise." [2]

While I agree with the above quote, I believe that one can learn some precursor signs of impending memory shortage. I have referred to these in numerous articles as "yellow" resource shortage indicators. With that in mind, let me illustrate this principle by using the results of some memory stress testing on a Series 807 HP-UX machine. The details relating to the test setup are the same as in the CPU case study above. For the following discussion please refer back to Figures 1-4.

It is intuitively obvious by looking at Figure 1 to expect response times to increase as more users are added onto the system. Furthermore, as the amount of main memory required by each process increases (in fact doubles) in figure 2, the response time stays the same for user tests 1 through 6. This is because the system has not yet reached memory saturation; i.e. it has not started swapping yet. At the 9 user, response times increase a bit more dramatically. For a memory load twice this value (1000) in Figure 3, user levels 9 and 12 represent quite a large increase in response times (in fact, a jump by a factor of over 10 times). The question is, what memory metrics reveal this? How could we have caught this earlier when the problem was in the yellow zone, rather than in the red zone like in the 12 user test?

One key will help by looking at Figure 9. In Figure 9 we see four memory "pulse points" illustrated. While these are not the only memory metrics available, these are four that we graphed together in order to determine which correlations might exist between such indicators. While it is clear that transaction response times increased a bit in Figures 1 and 2, why the dramatic jump(s) in Figures 3 and 4?

Figure 9 - Memory Test Resource Indicators

Notice that page faulting, along with page-outs, increase linearly up through the 9 and 12 user level. However, at the 9 and 12 user level, page faults literally go berserk and page out activity hovers between 10 and 20 per second. A page fault occurs when a necessary piece of data or code is not found in memory. A page fault under HP-UX is, in principle, the same thing as on an MPE system. The kernel will issue a disk I/O request in order to retrieve the missing piece(s). Some of this kind of memory action will normally occur even when there is no memory problem. A page out is a bit more serious than a page fault. It signifies that pages of data were literally having to be kicked out of main memory in order to accommodate more important needs. I have not used page

HP 9000-807 Memory Test; Load=1000
Memory Activity
Memory Activity Rates (Secs)
Read Hit%
100
90
80
70
60
50
40
30
20
10
0
1 User
3 Users
6 Users
9 Users
12 Users
20:10
20:15
20:21
20:26
20:31
20:37
20:43
20:48
20:56
22:08
10:18
10:23
10:29
Time (XM1L3122)
Page Faults
Page Outs
Swap Outs
Read Hit%

FIGURE 9:

outs on MPE systems as an indicator of memory stress. Since response times were relatively civil for the 1 through 9 users, we can loosley conclude that for page faulting less than 40 per second, the system probably has adequate main memory.

Using similar logic, we can conclude, for page-out actions, that less than 10 per second is safe. Note how page faults rise to incredible heights but page-outs stay steady within a range. There are often mechanisms within operating systems that prevent debilitatingly high levels of certain activities to occur. Page outs may be one of these actions. Page faults, on the other hand, go through the roof in this case study. Notice the oddity with page faults. When things get really bad (I'll show this in a moment), page faults actually taper off. So, it seems from this that we can conclude that page faults may be a reliable indicator of a memory shortage only prior to an actual crisis. This is actually a blessing in disguise. What we really need is a "yellow" zone indicator. Page faults (>=40 per second) seem to do the trick, at least in this case study on this 807.

Figure 9 seems to show a zero swap out rate for the duration of all the experiments. A swap out is a serious event that involves kicking all the pieces of a process out to disk. I know of no MPE equivalent of this activity. In figure 9, due to the low granularity of the right Y-Axis scale, you cannot see the very small amount of swap-out activity that is actually taking place beginning at 20:48. Refer to figure 10 to see this event more clearly.

Figure 10 - Memory Indicators - Swap Out Correlative

For figure 10 I changed the Y-Axis scale so that swap outs could be unveiled. Figure 3 shows a radical increase in response time for the 12 user scenario, while figure 10 reveals an ever so slight amount of swap-out activity. Some swap-out activity will take place for jobs that sleep for more than 20 seconds. This keeps memory from becoming cluttered. But, when memory becomes scarce, the system will perform desperation swapping (the "de" on the vmstat output) as seen above in figure 10 starting at 20:48. This action persists for the duration of the 12 user test. Since a swap-out occurs only in the case of a serious memory shortage, its consistent presence is a strong indicator that you either need to reduce your memory consumption or buy more!

This memory discussion should at least help you in monitoring your system's memory usage. Try to focus on the yellow levels of each indicator. Remember, for swap-outs, just about any will be a bad sign.

Conclusion

I hope that you have gleaned a little bit of insight into some of the CPU and memory key performance indicators that I have presented here. The case studies should be a good starting point if you are trying to understand and better manage your HP-UX system. While I have not focused on tuning aspects in this paper, that is the next logical step (perhaps the focus of a future article) in making the most of your company's hardware investment.

References:

[1] Lund, Robert A. "Taming the HP 3000 - Volume II - The Theory and Practice of

HP 9000-807 Memory Test; Load=1000
Swap Out Correlative
Memory Activity Rates (Secs)
Swap Out Rate (Secs)
100
90
80
70
60
50
40
30
20
10
0
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1 User
3 Users
6 Users
9 Users
12 Users
20:10
20:15
20:21
20:26
20:31
20:37
20:43
20:48
20:56
22:08
10:18
10:23
10:29
Time (XM3I3135)
Page Faults
Page Outs
Swap Outs

FIGURE 10:

Successful Performance Management for Hewlett-Packard HP 3000 Computer Systems." (Albany, Oregon, Performance Press Publishing, 1992, (503) 327-3800), pg. 29.

[2] System Performance Tuning - Mike Loukides - (Sebastopol, Calif., O'Reilly & Associates, Inc., 1991), pg. 83.

UNIX Internals - Myril Shaw and Susan Shaw - McGraw-Hill 1987

About the Author: Robert Lund is the president of Lund Performance Solutions, which specializes in system performance software, consulting, and training for HP 9000 and HP 3000 computer systems. Robert is also the author of Taming the HP 3000 Volumes I and II (The Theory and Practice of Successful Performance Managment). His company has designed and developed major software packages including, SOS/9000 Performance Advisor, SOS/3000 Performance Advisor, Q-Xcelerator Resource Manager, Forecast Capacity Planner, and Performance Gallery - a Windows-based performance analysis and graphing package. Robert has been teaching seminars in the U.S. and Europe since 1988 and is considered an expert in the field of computer system performance.

Paper No. 6017
Do We Need Transaction Monitors On Open Systems?

Rolf Brandt
INFOSOFT GmbH
Hauert 12 a
44227 Dortmund
Germany
Phone: + 49 231 758 98-0

Operation and Functions of Transaction Monitors on Mainframes

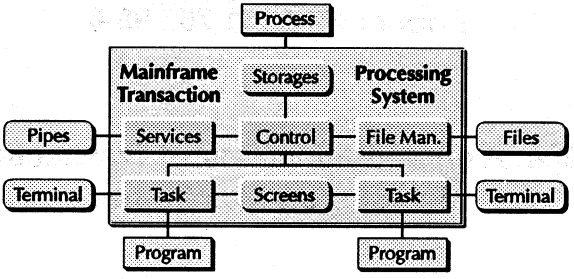
Mainframe users are accustomed to implementing on-line applications with the aid of transaction processing systems or transaction monitors. Without such a transaction monitor, no reasonable on-line processing is feasible. That is why mainframe users considering migration to Open Systems often ask whether a transaction monitor is also needed on these systems or which facilities are going to replace it. In order to answer to this question, first of all, we have to deal with the mode of operation of transaction monitors on mainframes. Then, we will look at the functions they usually accomplish.

As mainframe operating systems are not allowing direct on-line transaction processing, a special transaction control program (TCP) must be used for it. The TCP, like any other application program, is running as one process of the operating system but one with special characteristics for the necessities of on-line transaction processing. An event oriented interrupt control in a separate computer partition belongs to these characteristics in particular.

Operating with Task Control

For each defined user, the TCP creates a task - a quasi process in the process - assigning an appointed terminal to it. On the terminal, the user may type in the identification code of a transaction which is carried out subsequently. At its end, the transaction calls another transaction or returns control to the terminal.

A transaction consists of one or more TCP application programs representing sub-programs of the TCP and calling on various services of the TCP using special commands. The application programs are loaded into storage only once and are re-entrant in threads.



Among the services performed by the TCP, we find the task control, the screen processing, the file processing with locking and logging, the provision of transaction surviving storage areas, the communication between different tasks and TCP processes, the provision of internal TCP information, the recovery facility, and various administrative functions.

Benefits of Transaction Monitors

TCP's are very complex programs taking up considerable parts of main storage and processor performance. However, they are also said to have the capability of essentially improving the computer performance concerning on-line transaction processing. In particular, they are allowing to connect a large number of active terminals. The secret lies in better exploitation of the available computer capacity and resources by the TCP.

It is not necessary to build up a separate process environment for each user but just a task environment consuming less resources. It is neither necessary to separately open the file management system for each user. As all application programs are sub-programs of the TCP, it is knowing all users and the processing steps these users are just executing.

Event Controlled Interrupts

As a rule, an application program begins with receiving a screen and ends with sending the next screen. In between, it only returns control to the TCP in case any services are needed and requested from the control program. Only in the course of these interrupts and screen editing, an other user is getting a chance to proceed.

Screen processing is performed using block mode and terminals supporting this mode. The TCP is sending a data stream containing all information about the screen and the necessary control character sequences. If the screen has already been sent out before, just modified information is transmitted. The terminal only returns the values of the input fields as far as they have been changed. In the meantime, practically no processor time is needed for screen processing because it has just to be checked if input values are to be received.

Sequentialization of File Access

The **TCP** sequentializes file access in order to avoid deadlock situations and unwelcome interim data changes. Sequentialization is achieved by the circumstance that the file manager of the **TCP** does not accept any longer modification and locking orders regarding the files to be updated from other users after the first modification or explicit locking order of an user, as long as the lock is not raised explicitly or by transaction end.

If program crashes occur, the **TCP** can perform a dynamic recovery rolling back the file system to the latest attained consistent condition. In this case, program control returns to transaction begin or branches to an other checkpoint.

Data Transfer Provisions

The **TCP** provides different mechanisms passing on data to the next transaction, to an other user, or to other **TCP** processes. These provisions

range from storage areas within the **TCP** through external main storage and file areas to queues with different access types and pipelines. Updating of main storage areas being available to several users, directly affects all users immediately.

Additional Functions

Further on, the **TCP** administrates tables containing information on the relation of application programs to transactions, on files used in the application, and on other application characteristics. If required, it will trace information on its activities. Also, he **TCP** permits dynamic application changes.

Are Traditional Transaction Monitors Needed on Open Systems?

This question is often asked. It may clearly be answered with *no* if a client/server approach is chosen on Open Systems or even a heterogeneous distributed architecture is preferred. In this case, on the contrary, the functionalities of transaction control systems are required which are implemented on the base of the Distributed Computing Environment (DCE) of the Open Software Foundation (OSF) and are especially tailored to distributed processing environments. But the traditional transaction control programs (TCP) being installed on mainframes do not provide such functions.

For a pure client/server solution, as an alternative, the basic products being offered for it may be used. They contain a subset of the functionalities which are required for generally distributed systems. That is why the question which was put at the beginning narrows to the issue whether the functions of a TCP are needed on non-distributed Open Systems, that means on standalone systems on which all application components - maybe except screen processing - are implemented.

The TCP was originally created in order to enable on-line processing on the batch-oriented mainframes at all. There were required facilities for the communication with screen devices in connection with block mode processing and an as ever optimal exploitation of the resources like proces-

sor performance, main storage, and processes being very limited at that time.

Over the time, this pure on-line orientation moved into the direction of on-line transaction processing. Functionalities like protection of data access through checkpoints and rollback were added, as well as various control, service, and administrative functions.

Terminals and Screens

It is apparent that no particular facilities for the communication with terminals are required on present Open Systems any longer because these functionalities are already embedded in the operating system. The screen processing may be effected with the aid of dedicated and efficient products of numerous vendors. However, as a rule, this processing is not performed in block mode any more, resulting in advantages and disadvantages. If the disadvantages prevail in special situations, a block mode simulation with PC's may be the solution.

Exploitation of Resources

While the bottleneck lies above all in the area of data access today, it was clearly located in the central unit in former times. That is why the TCP was implemented in such a manner that one process is serving all users through an internal process-in-process concept with code-sharing in threads. So, processes, main storage, and processor performance could be saved.

A separate facility for code sharing is no longer necessary on Open Systems as, here, the compilers already generate re-entrant code in connection with the operating system. Process environments need not to be

economized any more because sufficient main storage is available. All users can dispose of their own processes. Also, processor performance is not a bottleneck any longer, especially since parallel and dedicated additional processors are provided in many cases.

An essential characteristic of the **TCP** lies in the fact that it is runs in particular computer partitions with special processes. These processes underlie an event-oriented interrupt control. As, on the contrary, Open Systems are based on a timesharing concept, in this regard, the throughput advantage of mainframes for extreme applications (for example reservation systems) cannot be achieved.

Protection of Data Access

Even if the resources of the central processing unit are not necessarily limited any more today, this does not mean that we may deal with them without thinking. However, the applications have to be designed in such a manner that the resources are used economically where it appears reasonable. For example, when designing, it may make sense to split the applications into client and different (maybe common) server processes. Above all, we have to pay attention that no superfluous information are transferred during data access and no over-dimensioned buffer areas are built up in main storage.

Moreover, as far as the TCP protects the database through checkpoints and rollback mechanisms, no provisions from a particular transaction control system are needed. The available data management systems already provide all functionalities. If required, they additionally may easily be completed by application-specific routines.

Control and Service Functions

The basic techniques for the communication methods between transactions, tasks, and processes which the TCP supplies are already provided by the operating system on Open Systems, for example through shared memory and inter-process facilities. The same is valid for various system services.

As the application programs represent subprograms of the TCP due to the task concept, the TCP has to execute the formal call sequence control. Logically, it may only be influenced indirectly. That is why

many users have implemented an additional application-specific call sequence control directly under the TCP causing that the control functions of the TCP practically become meaningless. As a matter of fact, they could be dropped.

The administrative functions of the TCP are partially enforced by the monitor concept and otherwise largely superfluous (as for example various control tables). On the other side, they are partially useful facilities (for example supervising functions). But these may be replicated rather easily if they are not already covered in another form anyhow.

Final Remarks

The characteristics of Open Systems differ a lot from those of mainframes. The presently predominant advantage of the TCP, the ability of allowing an extremely large number of on-line users for very homogeneous applications, can not yet be achieved on Open Systems due to the exclusive timesharing concept. However, this advantage is actually exploited just very seldom.

As far as the other functions of the TCP are needed at all, the operating system and basic systems are at least providing techniques for their implementation. This implementation mostly may be designed more practically on Open Systems because the rigid ties of the TCP are absent.

Distributed Transaction Processing on the Base of DCE

The Open Software Foundation (OSF) has proposed a standard for the basic services needed for distributed processing. This standard is known and widely accepted as Distributed Computing Environment (DCE). Overlaying DCE is the Encina Toolkit from Transarc Corporation which provides extensions for distributed transaction processing as well as a higher application interface. The Toolkit assists in the implementation of transaction control systems which provide additional functions typical to monitors. For example,

both the Encina Monitor and the CICS/6000 products are implemented with the Toolkit.

DCE defines five different sets of basic services:

- **client/server communication**
- **resource location**
- **security services**
- **scalability functions**
- **resource sharing**

The client/server communication service of DCE is based on the Remote Procedure Call (RPC) concept. This handles the details of creating and managing computer connections, messaging protocols and processing code sections (multi-threading). The RPC's may be generated from local procedures and allow a single client to access many servers and a server to serve multiple clients.

Basic Services of DCE

The naming service of DCE provides location services for local and global resources independent from the different naming conventions of the connected systems. This service allows a client to request a service by its assigned name without knowing the exact location of a required server.

DCE supplies three basic services for security purposes:

- **authentication control • verifying the identity of a requester via a password**
- **authorization control • supervising which resources a user is allowed to access**
- **encryption service • ensuring that no other user can intercept data whilst being transmitted**

The scalability functions guarantee that a distributed application will work over networks of varying sizes, from a single LAN to world-wide WAN's. DCE addresses this through the concept of network cells, independently

administered domains within the network. The cells manage their own local communication and provide services for cell-to-cell communication.

The Distributed File System (DFS) supports resource sharing and allows users to share remote files as if they were local files. It also supports data caching mechanisms for improved performance.

Encina Toolkit and Monitor

The Encina Toolkit consists of the Executive and the Server Core components. The Executive provides the functions which are required for distributed transaction processing. In effect, it supplies an interface (Application Programming Interface - API) for specifying transaction boundaries, concurrent processing and exception handling.

The Executive also includes Transactional "C", an extension of the ANSI standard for transaction processing, and Transactional RPC (TRPC) which adds transaction capabilities to DCE's RPC and, from a programmer's perspective, is the same as RPC.

With the Distributed Transaction Service (TRAN), a two-phase commit support structure is provided which communicates with remote resource managers through TRPC or other communication facilities (e.g. Advanced Peer to Peer Communication - APPC).

In the Server Core, the following functions are located:

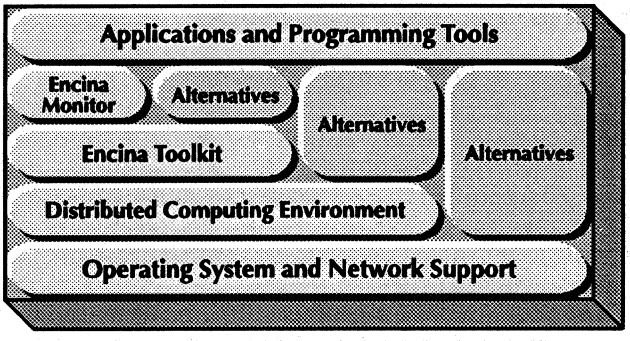
- **locking, logging and recovery**
- **interfaces for databases (XA from X/Open)**

The Monitor which is built on the Toolkit provides additional services:

- **configuration administration**
- **system monitoring**
- **security management**
- **performance control**

At present, only a few distributed transaction processing systems which are based on DCE monitors or similar facilities are used in corporate operations. This is apparently caused by the fact that few

distributed applications are currently installed and these applications are designed with the aid of less complex basic systems.



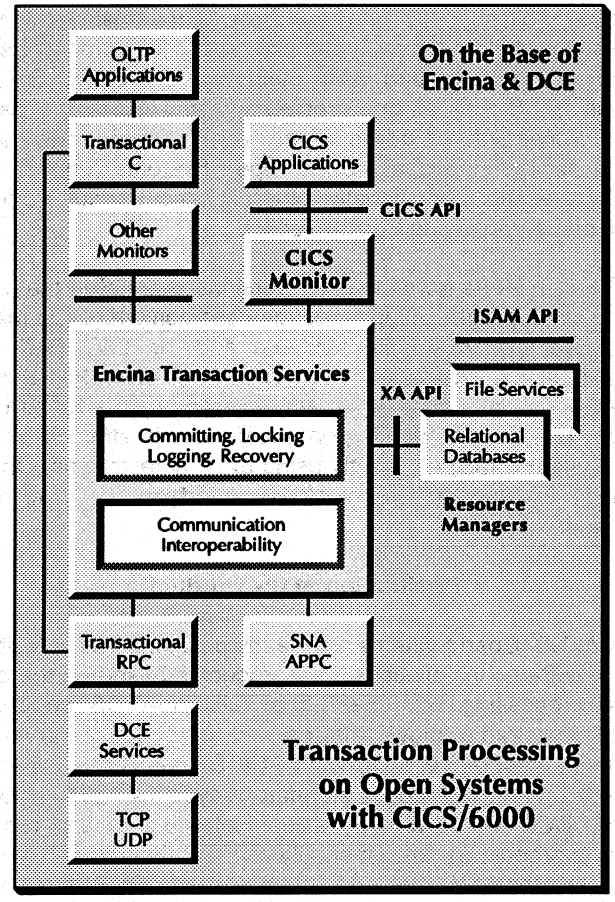
On the other hand, the technology can not yet be assessed as mature. There are missing links to traditional programming languages such as COBOL, fourth generation languages, databases and graphical user interfaces.

The use of DCE monitors on non-distributed, standalone systems does not appear viable with just a small set of the functionalities actually being required. More functions may be addressed, however, this may lead to system degradation without benefits.

Transaction Processing on Open Systems with CICS/6000

CICS/6000, a variant of the mainframe based transaction monitor CICS (Customer Information Control System) is either already available or will soon be available within the Open Systems environment on selected hardware platforms. CICS holds a leading position in the mainframe world and is used by over 22,000 organizations from 90 countries with 36,000 licenses. World-wide, over 300,000 programmers are currently utilizing CICS. Given this "critical mass" of expe-

rienced end-users, it makes sense to exploit industry familiarity with the product, by introducing the CICS functionality and methodology to the world of Open Systems in order to establish the product as the "de facto" standard.



CICS/6000 is not directly implemented under the UNIX operating system but is based on the Encina Toolkit from Transarc Corporation which, in turn, is based on the Distributed Computing Environment (DCE). The

Encina Toolkit provides an interface (Application Programming Interface - API) which is used by CICS/6000 for replication of its own CICS API and it is for this reason, that the CICS API is ultimately emulated with the aid of the Encina API. CICS/6000 has not just been ported from other CICS versions but is a fully autonomous product re-written in the "C" language.

The product itself supports only a subset of the mainframe version of CICS. This is partially caused by the different characteristics between the mainframe architecture and that employed by Open Systems. On mainframes, CICS runs as one process with multiple users participating within this process on an individual task basis. On the other hand, each user of the timesharing operation of Open Systems requires their own process for execution. Additionally, other limitations result from the characteristics of the underlying Encina Toolkit.

CICS/6000 also differs from the other versions of CICS in that it does not employ the same programming languages and compilers. On mainframes, CICS supports the VSAM file structure and the IMS, DL/I, DB2 and SQL/DS databases. Encina, however, supports its own File Services and various relational databases which are accessed by the XA API proposed by X/Open. The screen handling on mainframes is effected by the Basic Mapping System (BMS) or directly using 3270 datastreams. On the other hand, Encina will support Graphical User Interfaces (GUI) to handle the screen image.

Positioning of the Product

With CICS/6000, Open Systems are clearly positioned as satellite devices for CICS mainframe installations. The product is based on the concept of distributed processing with the aid of CICS-to-CICS communications where the mainframe plays the part of centralized server.For new developments outside the CICS world, CICS/6000 is not expected to be widely used. On non-distributed, standalone systems, the functionalities of Encina and DCE are to a large extent not required and represent an avoidable overhead in an Open System environment. However, for distributed environments, the use of Encina and DCE is advantageous and CICS/6000 will not greatly enhance their functionality.

When offloading mainframe applications, CICS/6000 provides limited benefits. On one hand, portation requires considerable effort because the batch portions of the application, the screen definitions

and the existing data need to be ported. On the other hand, migration of monolithic applications to separate basic components for a distributed processing environment makes little sense. However, if the applications are to be converted on a distributed basis, CICS is not necessarily required.

Paper Number 6019

Business Use of the INTERNET

Richard V.C. Tinker
Hewlett-Packard, Company
2015 South Park Place
Atlanta, Georgia 30339
(404) 988-3524

"The INTERNET is THE worldwide network." "TCP/IP IS the networking standard." "ALL networks join to the INTERNET." These are just some of the presumptuous statements that are heard every day when companies start discussing the INTERNET or their own networks. Recent changes by the National Science Foundation (NSF) could make many of these statements come true. Before I discuss the change NSF has made, let me first discuss what it is I am referring to: the INTERNET. When people refer to the INTERNET, they are speaking primarily about the government funded network that started it all. NSFNET is a network that was built many years ago for the government, educational institutions, and companies engaged in research for the government or schools. NSFNET was started by interconnecting or INTER-NETworking 19 computer nodes using very high-bandwidth data communication lines. IP, or Internet Protocol, was developed with this national network in mind. IP allowed the network to grow, it allowed for intelligent routing of information to the correct destination, and it provided a standard protocol for every entity connecting to the network. The IP protocol was designed to handle millions of addresses, and today, because of its enormous popularity, the INTERNET is running out of addresses. Task forces are scrambling to develop a fix for the addressing situation as I speak.

What is the value of having businesses, schools, and the government on the network? Businesses provide information on new technologies that are in use by schools, government, and their customers. Schools provide a wealth of information on what their PhDs have been up to, and the government provides what it is good at providing -- a great amount of electronically stored paper. What's even more special about the INTERNET is that it provides services too. It's more than just information retrieval - it's remote access to computing power, electronic messaging, technical discussions involving people from all over the world, and even the latest recipe for homemade Egg Nog if you want it.

One of the reasons this incredible network is not used by more companies has been NSF's stringent policy forbidding INTERNET use for commercial purposes. The following is taken from the NSF Acceptible Use policy: "...UNACCEPTABLE USES: (10) Use for for-profit activities, unless covered by the General Principle

or as a specifically acceptable use. (11) Extensive use for private or personal business. This statement applies to use of the NSFNET Backbone only...". It is very easy for an INTERNET connected business to send a message or connect to a customer in such a way that it inadvertently violates the acceptable use policy. This has been the limiting factor - a hard to enforce rule about usage that prevented many of our corporations from investing in the network and establishing connections to it. Well, guess what - it's all different today! The government's past policy was due in part to the fact that it was the taxpayers who were funding the network - not the commercial users. A decision announced in January of this year, which was more clearly defined in late May, has opened up the INTERNET for commercial use, provided that business picks up the tab and not the taxpayers. This is wonderful news when you consider that many companies were willing to pay for usage long ago - it just wasn't allowed until this year. A new service called the Internet Network Information Center, or INTERNIC, will provide the information users need to determine what INTERNET services could be of use to them. This new service combined with a new cost structure and usage policy will expand the size of the network very quickly, and according to different sources, the number of users of the network is already around 10 million!

As part of this new offering by the government, three new services are going to be started and administered by three companies: AT&T, Network Solutions, and General Atomics. AT&T is going to provide INTERNET directories, one of which will someday be roughly equivalent to producing a phone book for the world - I'd pay money to see the paper version of that torn in half by human hands! Network Solutions will have the arduous task of registering nonmilitary networks, and General Atomics is the new information reference provider for the network with their new INTERNIC information service.

The new INTERNIC team - if you use the INTERNET, you'll need this information:

Telephone: **1-800-444-4345** The telephone menu consists of option 1 for Registration Services provided by Network Solutions, Inc.; option 2 for Directory and Database Services by AT&T; option 3 for Information Services provided by General Atomics ; or option 4 for the Information Services Reference Desk or when you don't know which of the other three you want.

Electronic Mail Information and Requests: info@internic.net

INTERNIC Service Providers:

Network Solutions, Inc. Herndon, VA	*Phone:* (703) 742-4777 *Email:* hostmaster@rs.internic.net *FTP:* rs.internic.net
AT&T 5000 Handley Road Room 1B13 South Plainfield, NJ 07080	*Phone:* (908) 668-6587 *Fax:* (908) 668-3763 *Email:* admin@ds.internic.net *FTP:* ds.internic.net
General Atomics San Diego, CA	*Phone:* (619) 455-4600 *Email:* info@internic.net *FTP:* is.internic.net

General Atomics will provide pointers to services and information out on the network, and will even have people roaming the network in search of new services and resources. General Atomics and AT&T's part in this is very crucial to how companies will use the network since General Atomics is looking for new sources and AT&T is providing some of the references to them. If it is difficult and time consuming for a company to find the information or service they need, then it will be less willing to use the network service frequently, and may not get connected with a business that can provide what it needs. If this happens, then businesses depending upon INTERNIC to provide new customers will falter and stop vending their services. When that happens, companies close, jobs are lost, people stop spending money, the economy goes bad, airlines will raise their airfares, and President Clinton will be wondering what in the world started it all!

On a more serious note, it is interesting to see what the government's part in all of this has been. While their part of changing the policy was the most important, what made that happen? Could it be that Al Gore's interest in a nationwide network highway was suddenly elevated by his promotion from senator to vice-president or was it just overwhelming commercial demand? We may never

know the answer to that question. We now have an administration that is demonstrating a knowledge of what a nationwide network could do for us - and that helps. If high-level government backing continues, there's little doubt that the statements I made in the beginning will come true.

To get back to what the INTERNET could mean to your company, let's look at a few services that will demonstrate some possible uses of the INTERNET:

Exchange electronic mail messages with other people to -
- Schedule meetings with your customers and suppliers
- Take orders from your customers and send orders to your suppliers
- Answer questions about your products or services
- Announce new products or services

Send and receive files to -
- Update or track your inventories
- Update your customer's price lists or get new price lists from your vendors
- Distribute software releases and patches
- Circulate documents or files that others can work on
- Trigger a processing event to occur on a remote computer

Read UseNet News and connect to information services to -
- Keep abreast of technologies and information that effects your business
- Update your prices if they are dependent upon other industries or markets
- Watch your investments (for better or for worse) and...
- Sell that old lawn mower in the shed

Use services such as WAIS and GOPHER to -
- Find a reference source by electronically searching hundreds of library catalogs
- Pull information out of a database when the "push" information sources such as UseNet News have been exhausted

Any one of these features can mean a great deal of increased business for your company. Many of them could translate into a significant time savings for you and your customers. Depending upon how you use the INTERNET, the access cost could be more than offset by the increased business and time savings. Hewlett-Packard, Co. will often be asked to put on a demo of some new software for one of our customers - what a great convenience it is to be able to transfer over the software for the demo right from the HP division that wrote it! Imagine your company being able to make new software available to your customers the minute it's done being tested without having to cut a single disk or tape. If you run a research department, why waste employee time trying to find reference materials

when there are hundreds of different library catalog databases available over the INTERNET? If your local library participates in the library book exchange program, you can get a book you need sent to your local library in a matter of days.

One of the newer, really nice services being offered via INTERNET is called WAIS, or Wide Area Information Service. WAIS uses an intelligent method for disseminating a search string that you enter in English-like syntax, and then it goes out to as many machines as it has been told about to search for information regarding your request. The predecessor to WAIS was appropriately called GOPHER since like WAIS, it would "go-fer" what you wanted. Many larger companies have started putting their own company-wide documentation into private WAIS servers so that their employees would have a fast and efficient way to find the company related information they need. You could potentially even key in a search string for your son or daughter's research paper like "documents on King Edward" and watch it return data from the Medieval and Early Modern Data Bank archive at Rutgers, the State University of New Jersey on all of the King Edwards that have existed and all of the property they owned, provided the State University of New Jersey made their archive available via WAIS. The more creative and beneficial ways to use the network for you or your company depend upon your situation and how important it is to you - as you may have guessed, nothing is for free, but some of the best buys on data and services can be found on the INTERNET.

Just what am I talking about when I speak of this "cost thing"? Let me give you some idea of what to expect for connectivity to the network and services offered via the net. In May of this year, I called somebody at our corporate offices and asked him where some of our INTERNET connections are and who provides them. Two of our connections are with two of the original 19 regional networks that started it all. I was informed that we had just finished an agreement involving our commercial use traffic entering the network through those connections. One of the other big connections is through a commercial network service provider that had already built their own network around the INTERNET. While this particular commercial service provider has INTERNET connections, the main reason they built their own network around the INTERNET was because of the policy regarding commercial use. They were able to offer INTERNET-like connectivity between companies and their customers without using the real INTERNET network. Today, they still offer their network service, but it is probably used more for point-of-presence connectivity instead of cross-country network bypassing. Using information I gathered in May, here is what a few of the services cost: E-mail access runs anywhere from a flat fee of $50/month to hourly fees in the $3/$4 range depending upon modem speed. Add anonymous FTP, which is file transfer capability, and UseNet News for anywhere around $30/$40 per month. If

you want to appear on the network as a full-time node instead of connecting only when you need to, some companies can offer that service for less than $200 per month for a 9600 baud connection. For anywhere from $200 to $300 per month, you can even interconnect your company's entire LAN to a service provider with INTERNET connectivity.

Accessing the INTERNET can be easier than you think. If your company already has a network in place, then it may be possible for large numbers of your employees to make use of INTERNET features very quickly. If you only want a single PC or UNIX workstation to have the connectivity, then it is easier since there are software packages available for many hardware platforms using dial-up modem connections. Without getting into detail, let's take a look at how you might begin making a connection to the INTERNET. As mentioned earlier, if you desire only a single connection at dial-up modem rates, then there are several service bureaus around that will not only provide the connection, but can also provide all of the software you need to use it. If all you want is the capability to send messages and some small files to a supplier or customer, look into companies offering mail services with INTERNET connectivity such as AT&T Mail, MCI Mail, Compuserve, America On-line, Performance Systems International (PSI, Inc.), and a host of others. The minute by minute connectivity costs plus INTERNET mail costs may be quite reasonable depending upon how important it is to your company. Connectivity to the network on a much larger scale is what is often done by very large companies such as Hewlett-Packard, AT&T, IBM, and Digital. Connectivity on that scale gets a bit complicated, so let me give you some insight into what Hewlett-Packard uses for a corporate-wide network and the scope of our inter-connectivity.

When I was directly involved with networking at Hewlett-Packard, I was involved in building one of the largest, if not THE largest, privately owned routed LAN networks. At Hewlett-Packard, we call the network INTERNET too, but for sake of clarity, most people will designate our network as "Little I" INTERNET (or HP INTERNET) and the worldwide INTERNET as the "Big I" INTERNET. Several of our offices that I provided network connectivity to were offices of 50 people or less. While the main focus was connectivity for our internal data processing systems, my team was providing a type of INTERNET connectivity that the technical users in the office got all ecstatic and happy about. I say that the connectivity I installed was a type of INTERNET connectivity, but it was really connectivity into HP's INTERNET which is heavily secured from the "Big I" INTERNET. It is not possible for an HP employee to connect directly to one of NSF's service machines, just as it is not possible for an employee of DEC to connect to one of our machines. We do provide access to the "Big I" INTERNET in different ways though so that it remains an easy to use, and productive source of information and service to our employees and customers. For this type of

connection, all of the computers that are going to communicate on the INTERNET will need to use IP (usually TCP/IP) as its network protocol, or will at least have to encapsulate another network protocol within IP for transport over the INTERNET, because it is an IP based network.

There are many vendors that provide TCP/IP protocol stacks for many different hardware platforms - I know of at least 8 to 10 for the DOS PC platform. Because of its popularity, TCP/IP networking or at least TCP/IP encapsulation is being offered by probably every one of the top 20 computer manufacturers. If there is a top 20 computer manufacturer that doesn't provide TCP/IP for its machine, then keep watching as it slowly slips out of the top 20 category!

When you implement IP on your LAN, you'll have to learn about IP addresses, subnet masks, IP classes, and other important information before you are able to interconnect the LAN to anything else. When a good understanding of IP is established, go forward and request an IP address class and range from the company handling registrations which is Network Solutions, Inc. After getting an IP address assignment, learn enough about subnetting so that you can further break down the IP addresses and start planning for your own company's growth. Subnetting, or breaking the IP address range into different networks, is also the best way to interconnect different sites within your company. Another piece of the registration process is the domain name. As you learn more about the network, you will realize that memorizing four numbers to get to a machine is not always easy or convenient. Domain Name System (DNS) is the software that allows a user to key in a machine name and have other computers on the network do a lookup of the IP address that corresponds to that name. The full name of a computer on the network consists of two parts: the node or machine name followed by the domain in which it resides. For instance, John Doe's personal workstation is called JDOE, John works for Hewlett-Packard in Atlanta, GA, so his full machine name is JDOE.ATL.HP.COM. If we break down the meaning of the letters between the periods, we find a top level domain of COM which means COMmercial network, HP is the domain for Hewlett-Packard, Co., ATL is a sub-domain created by HP to indicate a machine in Atlanta, GA, and finally JDOE is the name of the machine itself. Some of the useful top level domains to know include COMmercial, MILitary, GOVernment, EDUcation, and NETwork. When somebody is trying to connect to John's machine and references it by name, their computer (providing they use DNS) will generate a message for the local network machines looking for its own domain name server. When the domain name server is found, a repetitive series of events occurs until the IP address is found or until an "authoritative" server for the domain responds with the equivalent to "I don't know about it, so it must not exist"! The repetitive series of events that occurs is that the first domain name server will look in it's own memory to see if it has the IP address because somebody else requested it before; if the name is not found, then it goes to

a higher level authority server and asks it for the address. This process repeats until the highest level domain name server is queried. The highest level domain name server would be the one authoritative for the COM, MIL, EDU, NET, etc. domains. During this search process, if the request comes across a domain name server that is the final, or "authoritative" server for the domain and it doesn't know the IP address, then the search is canceled with a "not known" result. DNS is very powerful and is a good reason why Network Solutions, Inc. will be asking you for one when you request an IP address.

The one nifty little communications device that hasn't been talked about yet that makes much of this possible is the router. As the name implies, it "routes" LAN packets depending upon the IP addresses contained within it and how much it knows about the network. When a router is connected to another network, it gathers information about the other network and starts building tables of routing information so that it can send LAN packets not only in the right direction, but also along the path that will get it there the fastest or at the lowest cost - depending upon how it is set up. Routers have one or more LAN interfaces, and are usually available with the LAN interface matching what your LAN consists of. Routers that interconnect remote networks will also have some sort of interface that allows the router to talk to another router over a data communications link. Some routers talk to simple 2-wire dial-up modems, others to T3 speed digital service units. Once a router has been set up to connect two networks, it "routes" the LAN packets that want to go to another network and leaves the packets destined for the local network alone. Routers can be pretty expensive, but one must consider how much work they do and how important and integral they are to the routed LAN network. Many times it is the router that manages security between private network and the INTERNET - this issue makes the router an important piece of equipment in the eyes of your internal auditors too. Because routers can be expensive and time consuming to maintain, many network service providing companies have options available where they provide and maintain the router. This service may be the one that is right for your company if you don't want to include the cost of a network engineer and technician on your next payroll.

Digest all of the information provided in this document, and what you find is that even the smallest of companies has INTERNET potential without a large investment of people or money. Refer to **Appendix A** for a list of Network Provider Referrals that was provided by the NSF Network Service Center (NNSC), and to **Appendix B** for a brief bibliography of articles on introductory level networking.

Let me now summarize the steps to take to obtaining INTERNET connectivity with some additional comments on each one:

1. Gather the information about the machine or network you want to connect to the INTERNET.

It is important to know things like how many machines you currently have and how many you plan to have in the near future. It would be a colossal waste of resources if you obtained a range of IP addresses for your network and ran out of IP addresses in a short time - the additional range you get may not be consecutive with your first range which could make further divisions of the address range difficult for your company to manage. The last thing you want to do is to have to ask an office with 100 machines in it to change their IP addresses - it turns even the most mild mannered employees into a bunch of raging lunatics! The type of network usage you plan to have is important too. If you are going to primarily use the network for electronic mail, then low bandwidth connections at a few strategic points may be all you need whereas interactive usage such as file transfers and remote machine access will require many high-bandwidth lines to get good performance.

2. Learn enough about IP addressing to be dangerous and then call Network Solutions, Inc. and ask for IP addresses and establish your domain.

Remember to plan for subnetting if you are connecting a very large network or if you are connecting multiple sites within your own network to the INTERNET. Obtain enough IP addresses to plan for future growth without hogging up the limited amount that are left.

3. Start configuring machines on your network and getting it ready for TCP/IP connectivity.

For obvious reasons this can be a minor task for three personal computers in a dentist's office or a major long-term project for a 1000 node, 20 city, multiple platformed company. You may need to have the routers or computers do conversions from your network protocol to TCP/IP, or you may choose to completely convert to TCP/IP in phases.

4. Use the network knowledge obtained earlier to select a network service provider that meets your needs.

This is where cost becomes a real issue. Your options are: a) provide all equipment and datacomm lines yourself and just pay for connectivity, b) pay for equipment, datacomm lines, and connectivity but have a network service provider manage the whole ball of wax or c) pay to have a network service provider supply all equipment and manage all aspects of the connected network. Whatever your

part in the network connectivity is, remember to plan for support costs and include costs to feed the network and foster new growth. Remember, the dog doesn't bite the hand that feeds it, but you never hear what it does when it's left hungry!

5. Use General Atomics and resources out on the network to find new, innovative, and useful ways to use the network.

The network can become a means for employees to sit for hours aimlessly wandering net-land with bleary, blood-shot eyes just as easily as it can empower salesmen and support personnel with the ability to generate additional income for the company. Make the resources on the network work for you or use the network to give your customers an entirely new level of support. You'll be amazed by how much more information you can get from a client when the client can reply to your electronic mail at his or her leisure instead of having to play phone-tag with you for a week - the relationship building possibilities are tremendous. Using the network to your advantage, and finding good network based resources for your company is what it's all about and it's what you'll enjoy paying the invoices for each month.

The INTERNET isn't some new standard just adopted by a few computer manufacturers - it's the most popular, well known, and established network in the world. Because of sheer size alone, anything put in place or adopted for use on the INTERNET almost guarantees that it will become a standard. The organizations that have been forming it over the years have developed a solid foundation to build upon. With the addition of commercial use, it's quite possible you'll see INTERNET use in everybody's home before ISDN, interactive TV, video telephone, virtual reality shopping, or any other up and coming technology.

Appendix A

Network Provider Limited Referral List

This list is a combination of lists that were provided by the NSF Network Service Center - maintenance of this list is now done by the InterNIC. This list is the combination of the "Network Provider Referral List" and the "Limited Referral List: Network Providers for Low-Volume Users". The InterNIC can be reached at 1-800-444-4345.

This list is as of March 5th, 1993.

Dialup Service Types:

EM	-	Electronic Mail
IP	-	Dialup IP including TELNET, FTP and possibly other services
PDN	-	Access to Provider is available over Public Data Networks
U	-	USENET News
(n/a)	-	Information not available or not provided on original NNSC list (call provider for information)

Network Name / Contact Name	Service Area / Phone Number	Mail Address	Dialup Service
Providers Based in the United States of America			
Alternet	US and International		EM/IP U
UUNET	(800) 4UUNET3	alternet-info@uunet.uu.net	
Anterior Technology	California		EM U
Geoffrey Goodfello	(415) 328-5615	geoff@fernwood.mpk.ca.us	
AT&T MAIL	US		EM PDN
	(800) MAIL-672	postmaster@attmail.com	
ANS	US and International		(n/a)
Joel Maloff	(313) 663-7610	info@ans.net	
BARRNet	Northern/Central California		(n/a)
Paul Baer	(415) 723-7520	info@nic.barrnet.net	
CERFnet	Western US and International		EM/IP
CERFnet Hotline	(800) 876-2373 (619) 455-3900	help@cerf.net	
CICnet	Midwest US (MN, WI, IA, IN, IL, MI, OH)		EM/IP
John Hankins	(313) 998-6102	hankins@cic.net	
CO Supernet	Colorado (CO)		(n/a)
Ken Harmon	(303) 273-3471	kharmon@csn.org	
CompuServe	US and International		EM PDN
	(614) 457-8600		
CONCERT	North Carolina (NC)		(n/a)
Joe Ragland	(919) 248-1404	jrr@concert.net	
DIGEX	Washington DC, Baltimore, MD		EM/IP
Douglas Huphrey	(301) 220-2020	doug@digex.com	
FidoNet	US & International		EM
David Dodell	FAX: (602) 451-1165	hostmaster@fidonet.fidonet.org	
HOLONET	US & International		EM/IP U PDN
Arthur Britto	(510) 704-0160	info@holonet.net	

ICG	US & International		EM PDN
	(415) 442-0220	igc.org	
International Connections Manager (ICM) International			EM/IP
Robert Collet	(703) 904-2230	rcollet@icm1.icp.net	
INet	Indiana (IN)		(n/a)
Dick Ellis	(812) 855-4240	ellis@ucs.indiana.edu	
JVNCnet	US and International		EM/IP
Sergio Heker	(800) 35TIGER	market@jvnc.net	
Los Nettos	Los Angeles Area (CA)		(n/a)
Ann Westine Cooper	(310) 822-1511	los-nettos-request@isi.edu	
MichNet/Merit	Michigan (MI)		(n/a)
Jeff Ogden	(313) 764-9430	jogden@merit.edu	
MIDnet	Mid US (NE, OK, AR, MO, IA, KS, SD)		(n/a)
Dale Finkelson	(402) 472-5032	dmf@westie.unl.edu	
MRnet	Minnesota (MN)		(n/a)
Dennis Fazio	(612) 342-2570	dfazio@mr.net	
MSEN	Michigan (MI)		EM/IP
Owen Medd	(313) 998-4562	info@msen.com	
NEARnet	Northeastern US (ME NH VT CT RI MA)		EM/IP
John Curran	(617) 873-8730	nearnet-join@nic.near.net	
NETCOM	California (CA)		EM/IP U
Desirree Madison-Biggs	(408) 554-8649	des@netcom.com	
netILLINOIS	Illinois (IL)		(n/a)
Joel L. Hartman	(309) 677-3100	joel@bradley.bradley.edu	
NevadaNet	Nevada (NV)		EM/IP U
Don Zitter	(702) 784-6133	zitter@nevada.edu	
NorthwestNet	Northwestern US (WA OR ID MT ND WY AK)		(n/a)
Eric Hood	(206) 562-3000	ehood@nwnet.net	
NYSERnet	New York (NY)		EM/IP
Jim Luckett	(315) 443-4120	info@nysernet.org	
OARnet	Ohio (OH)		EM/IP PDN
Alison Brown	(614) 292-8100	alison@oar.net	
OMNET	US & International		EM PDN
	(617) 244-4333		
PACCOM	Hawaii (HI) and Australia, Japan, Korea, New Zealand, Hong Kong		(n/a)
Torben Nielsen	(808) 956-3499	torben@hawaii.edu	
PREPnet	Pennsylvania (PA)		(n/a)
Thomas Bajzek	(412) 268-7870	twb+@andrew.cmu.edu	
PSCNET	Eastern US (PA, OH, WV)		(n/a)
Eugene Hastings	(412) 268-4960	pscnet-admin@psc.edu	
PSINet	US and International		EM/IP U
PSI, Inc.	(800) 82PSI82	info@psi.com	
	(703) 620-6651		
PORTAL	US and International		EM PDN
John Little	(408) 973-9111	jel@corp.portal.com	
SDSCnet	San Diego Area (CA)		(n/a)
Paul Love	(619) 534-5043	loveep@sds.sdsc.edu	
Sesquinet	Texas (TX)		(n/a)
Farrell Gerbode	(713) 527-4988	farrell@rice.edu	

SprintLink	US and International		(n/a)
Bob Doyle	(703) 904-2230	bdoyle@icm1.icp.net	
SURAnet	Southeastern US		(n/a)
	(WV, VA, SC, NC, TN, KY, LA, MS, AL, GA, FL, DC, MD, DE)		
Deborah J. Nunn	(301) 982-4600	marketing@sura.net	
THEnet	Texas (TX)		(n/a)
William Green	(512) 471-3241	green@utexas.edu	
VERnet	Virginia (VA)		(n/a)
James Jokl	(804) 924-0616	jaj@virginia.edu	
Westnet	Western US (AZ, CO, ID, NM, UT, WY)		(n/a)
Pat Burns	(303) 491-7260	pburns@yuma.acns.colostate.edu	
WiscNet	Wisconsin (WI)		(n/a)
Tad Pinkerton	(608) 262-8874	tad@cs.wisc.edu	
World dot Net	Pacific NW (OR, WA, ID)		(n/a)
Internetworks, Inc.	(206) 576-7147	info@world.net	
WVNET	West Virginia (WV)		(n/a)
Harper Grimm	(304) 293-5192	cc011041@wvnvms.wvnet.edu	

Providers Based in Canada

ARnet	Alberta		(n/a)
Walter Neilson	(403) 450-5187	neilson@TITAN.arc.ab.ca	
BCnet	British Columbia		(n/a)
Mike Patterson	(604) 822-3932	Mike_Patterson@mtsg.ubc.ca	
MBnet	Manitoba		(n/a)
Gerry Miller	(204) 474-8230	miller@ccm.UManitoba.ca	
NBnet	New Brunswick		(n/a)
David MacNeil	(506) 453-4573	DGM@unb.ca	
NLnet	Newfoundland and Labrador		(n/a)
Wilf Bussey	(709) 737-8329	wilf@kean.ucs.mun.ca	
NSTN	Nova Scotia		(n/a)
Michael Martineau	(902) 468-NSTN	martinea@hawk.nstn.ns.ca	
ONet	Ontario		(n/a)
Andy Bjerring	(519) 661-2151	bjerring@uwovax.uwo.ca	
PEINet	Prince Edward Island		(n/a)
Jim Hancock	(902) 566-0450	hancock@upei.ca	
RISQ	Quebec		(n/a)
Bernard Turcotte	(514) 340-5700	turcotte@crim.ca	
SASKnet	Saskatchewan		(n/a)
Dean C. Jones	(306) 966-4860	jonesdc@admin.usask.ca	

Other Providers

AARNet	Australia		(n/a)
AARNet Support	+61 6 249 3385	aarnet@aarnet.edu.au	
UKnet	United Kingdom of Great Britain and N. Ireland		EM
UKnet Support	+44-227-475497	postmaster@uknet.ac.uk	
EUnet	Europe, CIS-region, and Northern Africa		(n/a)
EUnet Support	+31 20 592-5124	glenn@eu.net	
Pipex	United Kingdom		(n/a)
Richard Nuttall (RN131)	+44 223 424616	sales@pipex.net	

Appendix B

Additional Sources of Information

Network Working Group
Request for Comments: 1463
FYI: 19

E. Hoffman
Merit Network, Inc.
L. Jackson
NASA

May 1993

FYI on Introducing the Internet--
A Short Bibliography of Introductory Internetworking Readings
for the Network Novice

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Abstract

This bibliography offers a short list of recent information resources that will help the network novice become familiar with the Internet, including its associated networks, resources, protocols, and history. This FYI RFC includes references to free sources of information available on-line as well as traditional publications. A short section at the end includes information for accessing the on-line files. This FYI is intentionally brief so it can be easily used as a handout by user services personnel.

Acknowledgments

This document is based upon the work of the User Documents Working Group in the User Services Area of the Internet Engineering Task Force (IETF).

Bibliography of Introductory Readings

The following list includes a number of sources for learning about the Internet. If you have more questions about the Internet, the best source of information is your network service provider. For those interested in finding out about getting Internet connectivity, the books listed will help you locate a network service provider.

* Items with an asterisk are available at no charge on-line on the Internet--see the final section for information on how to obtain these.

1a. Introductory Papers

* Kehoe, Brendan P. (1992) "Zen and the Art of the Internet: A Beginner's Guide to the Internet," (first edition) 95 p.

* Krol, E. and E. Hoffman. (1993) "What is the Internet?" 11 p. (FYI 20, RFC 1462).

* Malkin, G. and A. Marine. (1992) "FYI on Questions and Answers: Answers to Commonly Asked 'New Internet User' Questions," 32 p. (FYI 4, RFC 1325).

* LaQuey, Tracy with Jeanne C. Ryer. (1992) "The.Internet.Companion," 30 p. (on-line chapters from book published by Addison-Wesley)

1b. Introductory Books: Basic User Guides

Kehoe, Brendan P. (1993) Zen and the Art of the Internet: A Beginner's Guide, (second edition) 112 p. Prentice Hall, Englewood Cliffs, NJ.

Krol, Ed. (1992) The Whole Internet User's Guide and Catalog, 400 p. O'Reilly & Assoc., Inc. Sebastopol, CA.

LaQuey, Tracy with Jeanne C. Ryer. (1992) The Internet Companion: A Beginner's Guide to Global Networking, 208 p. Addison-Wesley, Reading, MA.

1c. Introductory Books: Connection Starters

SRI International. (1992) Internet: Getting Started, 318 p. SRI International, 333 Ravenswood Ave., Rm. EJ291, Menlo Park, CA 94025.

2. Internet services and resources

* Martin, J. (1993) "There's Gold in them thar Networks! or Searching for Treasure in all the Wrong Places," 39 p. (RFC 1402/FYI 10).

* Merit Network, Inc. (1992) "Cruise of the Internet," Merit Network Inc., Ann Arbor, MI. (Disk based tutorial available for Macintosh or Windows).

Metz, Ray (1992) Directory of Directories on the Internet, 175 p. Meckler, Westport, CT.

* NSF Network Service Center. (nd) "Internet Resource Guide," NSF Network Service Center, Cambridge, MA.

3. Internet networks

Frey, Donnalyn and Rick Adams. (1991) !%@:: A Directory of Electronic Mail Addressing and Networks, (second edition) 436 p. O'Reilly & Assoc. Inc. Sebastopol, CA.

LaQuey, Tracy L. (1990) User's Directory of Computer Networks, 653 p. Digital Press, Bedford, MA.

Quarterman, John S. (1990) The Matrix: Computer Networks and Conferencing Systems Worldwide, 746 p. Digital Press, Bedford, MA.

4. Introducing the Internet Protocols

Comer, Douglas E. (1991) Internetworking With TCP/IP: Volume I, Principles, Protocols, and Architecture, (second edition). 547 p. Prentice Hall, Englewood Cliffs, NJ

* Hedrick, Charles L. (1987) "Introduction to the Internet Protocols," 34 p. Rutgers University Computer Science Facilities Group, Piscataway, NJ.

Lynch, Daniel C. & Marshall T. Rose (eds). (1993) The Internet System Handbook, 822 p. Addison-Wesley, Reading, MA.

6. Further Reading

* Bowers, K. L. et al. (1990) "FYI on Where to Start: A Bibliography of Internetworking Information," 42 p. (RFC 1175/FYI 3).

* Malkin, G. & T. LaQuey Parker. (1993) "Internet Users' Glossary," 53 p. (RFC 1392/FYI 18).

Getting Articles On-Line

All the documents marked with an asterisk (*) in this bibliography are available on-line at no charge if you have access to the Internet. To find out more about accessing documents in introducing.the.internet, send electronic mail to: nis-info@nic.merit.edu, with the text: send access.guide.

If you know how to use Anonymous FTP, you can get the Access Guide from one of several sites, including nic.merit.edu, nic.mr.net, ftp.nisc.sri.com, or ftp.hawaii.edu. Check the directory introducing.the.internet for the file titled access.guide. The Access Guide includes information about obtaining documents through dial-up service using a modem for those who do not have direct Internet access.

In addition to the listed publications, many network providers publish excellent user guides and newsletters which cover Internet topics. Contact your Internet network service provider for more information. A longer bibliography with more comprehensive references and abstracts, FYI 3, RFC 1175 is listed above for those who may need more detailed materials.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Ellen S. Hoffman

Merit Network, Inc.
2901 Hubbard, Pod G
Ann Arbor, MI 48109

Phone: (313) 936-3000
E-mail: ellen@merit.edu

Lenore Jackson

NASA Ames Research Center
m/s 233-8 T-1191
Moffett Field, CA 94035

Phone: (415) 604-0455
E-mail: jackson@nsipo.arc.nasa.gov

Appendix C

NSFNET Networks by Country

June 1, 1993

(Acquired from: NIC.MERIT.EDU File: /nsfnet/statistics/nets.by.country)

Code	Country	Net Count	Initial Connection
AR	Argentina	1.	10/90
AU	Australia	402.	05/89
AT	Austria	95.	06/90
BE	Belgium	28.	05/90
BR	Brazil	73.	06/90
BG	Bulgaria	6.	04/93
CM	Cameroon	1.	12/92
CA	Canada	522.	07/88 *
CL	Chile	16.	04/90
CR	Costa Rica	2.	01/93
HR	Croatia	2.	11/91
CY	Cyprus	8.	12/92
CZ	Czech Republic	122.	11/91
DK	Denmark	10.	11/88
EC	Ecuador	83.	07/92
EE	Estonia	13.	07/92
FI	Finland	128.	11/88
FR	France	595.	07/88 *
DE	Germany	574.	09/89
GH	Ghana	1.	05/93
GR	Greece	11.	07/90
HK	Hong Kong	9.	09/91
HU	Hungary	21.	11/91
IS	Iceland	8.	11/88
IN	India	3.	11/90
IE	Ireland	35.	07/90
IL	Israel	61.	08/89
IT	Italy	285.	08/89
JP	Japan	310.	08/89
KW	Kuwait	1.	12/92
LV	Latvia	1.	11/92

LU	Luxembourg	4.	04/92
MY	Malaysia	3.	11/92
MX	Mexico	37.	02/89
NL	Netherlands	144.	01/89
NZ	New Zealand	91.	04/89
NO	Norway	60.	11/88
PL	Poland	42.	11/91
PT	Portugal	35.	10/91
PR	Puerto Rico	4.	10/89
RO	Romania	1.	04/93
SG	Singapore	28.	05/91
SK	Slovakia	13.	03/92
SI	Slovenia	13.	02/92
ZA	South Africa	63.	12/91
KR	South Korea	38.	04/90
ES	Spain	59.	07/90
SE	Sweden	124.	11/88
CH	Switzerland	100.	03/90
TW	Taiwan	165.	12/91
TH	Thailand	14.	07/92
TN	Tunisia	1.	05/91
TR	Turkey	12.	01/93
GB	United Kingdom	467.	04/89
US	United States	7398.	07/88 *
VE	Venezuela	6.	02/92
----		-------	
56	Total	12349	

* Merit began managing the NSFNET backbone in July, 1988.

Paper Number: 6020

HOW TO MANAGE CHANGE WHEN MAKING A MAINFRAME ALTERNATIVE DECISION

Peggy Parskey

Hewlett-Packard Company
5245 Pacific Concourse Drive Suite 100
Los Angeles, CA 90045
(310) 535-2701

As Hewlett-Packard consultants work with organizations, both inside and outside of HP, they are discovering that the promises of technology, particularly in the Mainframe Alternative (MFA) arena, are not always realized.

If the impartial observer is looking for a convenient scapegoat for the shattered expectations, the easy place to point is at the hardware vendor. But, upon closer scrutiny, the observer finds that the hardware and the software and the associated technology at all levels within the Information Architecture do indeed measure up to the promise. Everything works as specified, all the bells and whistles truly exist today and the capabilities and described flexibility are indeed as robust as the vendor represented.

So where else does our observer look? If the products that can be touched and felt aren't the problem, where else might one point the finger? Perhaps the problem is lack of training. But, when the microscope is focused on the project again, our observer discovers that upper management listened to the vendor. They dutifully ordered training for their critical people; they have enabled them to take the training and get hands-on experience. Moreover, the company has contracted with the vendor to spend several days per week on-site to coach the employees through the technical maze of the new system.

So, if training is not the problem, what else is? Perhaps the project has not been managed well. Has a project manager been assigned? Has a detailed plan been developed, has its progress been tracked and have action items been duly communicated? Has a commitment for resources been made to the project? Yes, then what's wrong?

Why is the project proceeding more slowly than expected? Why are the employees complaining about the difficulty of the Mainframe Alternative transition? Why isn't the organization realizing the promised benefits of cost reduction, improved efficiencies or flexibility? The answer often lies in the very essence of the organization: its management, its culture, and the ability of the entire organization to manage change. What most organizations fail to do when

undertaking major technological change is to understand the true business drivers and the overt or covert forces resisting the proposed changes. In addition, most organizations do not adopt a repeatable, proven methodology for identifying and grappling with these issues. Instead, middle and upper management in most organizations assume that with the right hardware and software decision, with the right training and technical consulting services and with effective project management, the project will stay on track.

Think again.

EMPLOYEE EXPECTATIONS

What, then is really expected of the employees who are charged with moving from the Mainframe Environment to a distributed processing, client/server environment? What they are being asked to change goes well beyond the acquisition of a new knowledge base, and requires changes in their behaviors, beliefs, assumptions, identity and emotions (see Illustration 1).

Let's explore each one of these in a bit more detail:

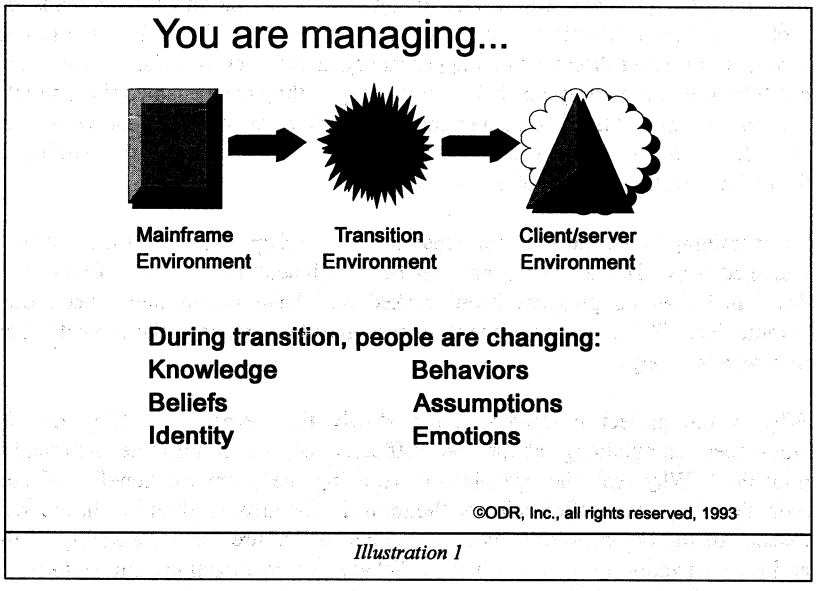


Illustration 1

Knowledge: Most companies introducing major technological change are well aware of the significant change in raw knowledge that must be acquired for the development, implementation and ongoing maintenance of the system to be successful. The CIO knows that he/she is introducing an entirely new operating system, new programming techniques, a new networking infrastructure, rapid application tools and a graphical user interface. To address these significant technical issues, the smart CIO identifies the critical training requirements and dutifully sends his/her people off to study.

Behaviors: At some level, most companies recognize that the behaviors of their IT employees must change in the new world order that is being created. After all, new technology is not only being introduced; in effect the organization is introducing an entirely new way of doing business. It is no longer sufficient for the program development people to remain closeted behind IS walls. They now need to be out mingling among the users, understanding their requirements and developing strategies for enhancing the business. IT is now getting more visibility than it ever did; they must be conversant both with the business issues and the enabling technology.

But, given these fundamental changes, what changes in performance measurement are introduced in most companies to reinforce this new behavior? What positive consequences exist, for changing their operating style? What negative consequences exist for not "getting with the program"? Most managers deal with these issues after the fact, when commitments aren't met or the project just isn't proceeding as planned.

Beliefs: What strongly held beliefs exist within the employees? Do they feel that the Mainframe has been solid and robust and has continued to meet the expectations of the organization? Do they fundamentally believe that the power of the mainframe will be sufficient to meet the business needs of the company? These deeply held beliefs, often not directly articulated, can prevent the staff from jumping on board the MFA bandwagon.

Assumptions: How often do you hear the phrase: "No one ever got fired for choosing IBM"? Not so frequently any more, I'll bet. But this long held assumption is probably ingrained in the very psyche of the IT staff and exorcising it is no easy task.

Identity: If the employees have been recognized for their ability to bang out code, what will be their worth when the mainframe disappears? If they were valued for their connections with third party suppliers of mainframe solutions, where will they be in the pecking order when those connections are no longer critical to the success of the systems operation? If their power base came in part from their leverage with the big mainframe vendor, what happens when the

vendor is no longer the top banana with your company? The identity of the individual contributors has been connected with a mainframe; reestablishing that identity in the client/server world will take time.

Emotions: If the software developers got their "psychic" rewards from developing superbly functioning code, they must now learn how to feel gratified from being a savvy business person. If all of their kudos have been for their technological acumen, it is not going to be easy to change their internal psychic reward process. Moreover, many employees feel that they may not have what it takes to function in this new environment. In coping with the impending change, they need to grapple with the impact these changes may have on their own careers.

****

The point of discussing this "warm and fuzzy" stuff is to identify the basic employee concerns that are rarely if ever considered in the transition process. Instead, management deals with the "hard stuff" and hopes that the transition will take care of itself.

What then can a CIO do to manage the transition more gracefully? How does he/she ensure that the implementation effort occurs in the time allotted, within the budget specified and with the full support of the implementation team?

The answer lies in the principles of Change Management. These principles have been developed over years of study of organizations that have managed change both successfully and unsuccessfully. One company that has developed an extensive body of research studying organizations is ODR®. Through their research they have concluded that organizations can be classified into the "Winners" and the "Losers". The "Winners", of course, have been able to meet budget and time constraints and have had successful projects and satisfied employees. The "Losers" often had projects that failed miserably, or took longer, cost more and generally left the entire organization with a bad taste in its mouth. From this research ODR® has been able to identify basic principles that can be followed to manage the change process more successfully. For the remainder of this paper, I am going to describe those principles and apply them to the Mainframe Alternative decision.

SEPARATING THE WINNERS FROM THE LOSERS

There are nine critical steps that the winners follow during their change projects:

1. *They recognize that change is a process*
2. *They understand that pain drives change and consciously surface this pain within the organization*
3. *They develop a compelling remedy for the pain*
4. *They tailor the communication plan to the various constituencies within the organization*
5. *They identify the sponsors and their roles during the project*
6. *They surface and manage the resistance to the change*
7. *They consciously plan the transition*
8. *They execute the plan*
9. *They monitor the change and make the necessary course corrections*

1. CHANGE IS A PROCESS

The winners recognize that change is not an event, but a process. Thus, management understands that they are attempting to move the organization from the present state: the Mainframe Environment, to the desired state: the distributed processing, client/server environment. However, to get from here to there requires the organization to pass through a transition state. Unfortunately, the transition state is characterized by low stability, and high, undirected energy. Employees who have been tied to the old mainframe environment are wondering if they can "cut it" in this new paradigm. So, they tend to spend a lot of time at the water cooler, taking to their buddies about the good old days and hoping that this new approach doesn't leave them out on the street. Even if they were looking forward to this change, once in the thick of it, many employees discover that "this is hard work!".

Because the winners understand that the entire shift to Mainframe Alternative is a process and not an event, they put in place a transition plan that deals not only with the technology, but the people issues as well.

2. PAIN DRIVES CHANGE

The winners also understand that pain is the driving force for change. Unsolved problems or opportunities can supply the pain that drives an organization to abandon the status quo and make the necessary technological and/or organizational changes. However, the pain must be felt at all levels within the organization, and unfortunately, most companies fail to realize this. Why do organizations make a mainframe alternative decision? Is cost a driving factor? In many cases, the need to significantly reduce costs is the key business driver for

the technological shift. But how low into the organization is this driving force understood? Often, not low enough. The individual contributors may sense that the cost structure is untenable, but do they realize the implications of maintaining the status quo? Is the need for a faster development cycle and more responsive IT the driving force? How clearly is that need understood? Or, have middle and upper management been taking the heat, absorbing the pain and not passing it on?

3. DEVELOP THE REMEDY

The winning organizations recognize that surfacing the pain is necessary, but not sufficient. Once top management helps employees understand that the existing cost structure is untenable, or that flexibility of the development cycle is imperative for the organization, affected employees must believe in a single remedy to alleviate the pain of maintaining the status quo. Developing a clear remedy that is supported by the key constituencies is how the winning organizations ensure that the organization is pursuing a single agenda.

The difficulty is getting the "targets of the change" to identify the same remedy that may be obvious to top level management. Often, the recognition that MFA is the right solution can result from engaging employees in re-engineering projects that examine cross-functional processes and identify technology enablers. In many instances, the mainframe will not enable the process re-design and thus the solution becomes obvious. In other cases, involving the software development team in an analysis of user needs can identify solutions that break out of the mainframe box.

By developing the mainframe alternative decision with the involvement of lower level employees within the organization, top management can build a strong support base at the same time that the solution is being developed. This process will go a long way towards minimizing the potential resistance that will inevitably arise within the organization.

4. COMMUNICATE THE CHANGE

Most organizations work hard at communicating impending changes to their organizations. However, most executives spend little time crafting how the message will be delivered to the troops. In many organizations, the technology decisions filter down through the organization in a haphazard way and become grist for the rumor mill.

Instead of allowing the random process to apply, the winners have identified at a high level, the impact that the MFA decision will have on various people and their workgroups within the organizations. Their announcements are tailored to the various constituencies, separating the concerns of users from those of software

developers or operations. Management is able thus to convey an understanding of the impact that the MFA decision would have on the organization. This open communication enables all employees to assess the effect of the decision on his or her job. Moreover, successful organizations recognize that the MFA decision has consequences that individuals may not assimilate immediately. So, they incorporate into the process mechanisms for on-going dialogue and two-way communication.

5. IDENTIFY THE SPONSORS AND CLEARLY DEFINE THEIR ROLES

Most organizations have identified a key individual who is responsible for initiating the MFA decision. Typically, the decision is driven by the top IT professional in the organization. This individual is the initiating sponsor who makes the technological decision to proceed. Typically, he or she controls the budget and has the overall responsibility for the success of the project. The successful organizations recognize that sponsorship doesn't stop with the person at the top. The on-going success of the project depends on a cascading set of sustaining sponsors who will continue to drive the project forward when the going gets tough. These sustaining sponsors recognize that they must continue to articulate why the organization has made the decision to move off the Mainframe. They must maintain the on-going dialogue as their organizations ask questions about the project and how it will affect workgroups and individuals. They must ensure that the key individuals, middle managers and supervisors, are trained to fulfill their respective roles during the execution of the MFA plan. They must also identify the key individuals who will be the agents of the change, responsible for the day-to-day implementation activities.

6. SURFACE AND MANAGE THE RESISTANCE

Most IT managers understand that resistance to change is inevitable and not a sign that something is inherently wrong with their decisions or their employees. However, knowing that it exists and knowing what to do about it are very different beasts. To avoid dealing with these uncomfortable emotions, many managers deny it. "No resistance in my organization, no sir. My folks are welcoming this change." In many companies the term "resistance" might as well be a four letter word.

What many managers also don't acknowledge is that resistance can't be "surfaced" unless the change has been clarified and announced. Until the employees determine how the Mainframe Alternative decision affects them personally, they will not resist. Of course, the rumor mill alone can get people spending lots of time speculating over the coffee machine, but not truly resisting.

Finally, employees can resist change even if it is welcomed. Once the actual MFA implementation is underway, many of the employees will discover that the

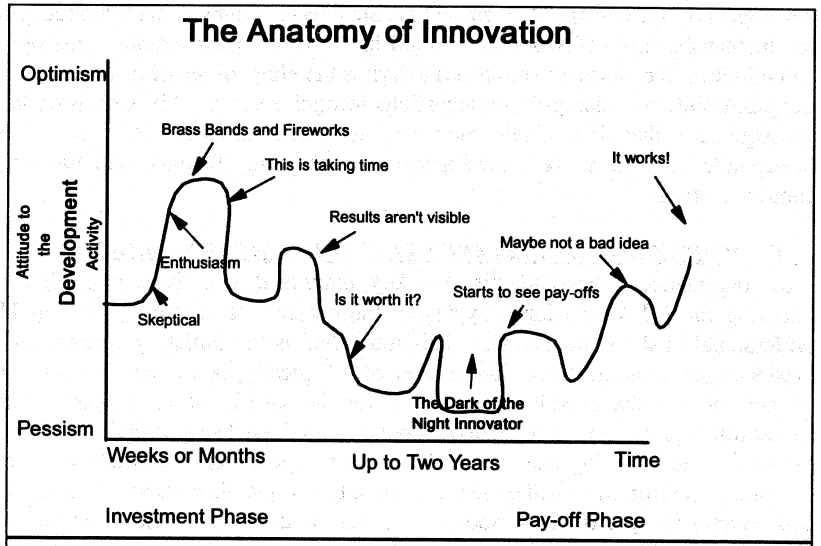


Illustration 2

transition was harder than they thought. Uninformed optimism can yield quickly to informed pessimism once the reality of the implementation begins. (See Illustration

2.) "I thought Unix would be a lot easier to learn!" "My old software development techniques aren't working with the distributed computing systems; what do I do now?" This discomfort with the transition process can manifest itself as strongly as resistance that comes from the employees who didn't want the new client/server stuff in the first place. It is important for the MFA implementation team to surface this resistance, and deal with it. People need to be allowed to vent, to get their feelings and frustrations out on the table and work out solutions.

7. PLAN THE TRANSITION

Most companies do an excellent job of planning the technical implementation for the migration to distributed systems. However, when the project falters, these organizations tend to blame the employees. It is not uncommon to hear: "My folks just don't have what it takes to function in this new environment."

The key point is that while the technological transition is important, there must also exist a parallel change management or people transition plan. This plan must identify the critical sponsors and the agents, and the roles and responsibilities they will have during the transition process. The sources of resistance must be clearly understood; a plan must be developed for coping with the resistance rather than

squashing it and hoping that it goes away. The culture of the organization must be realistically assessed. If the MFA solution is on a collision course with the existing culture, a plan must be developed for overcoming this barrier. In many organizations, the MFA transition is not just a technological issue, but a cultural one as well. If the entire operating philosophy will undergo a change, top management must develop a plan for effecting a cultural shift .

8. EXECUTE THE PLAN

The execution of the change management plan must dovetail with the execution of the technology implementation. For example, if the MFA move will cause the development teams to be reorganized, then dealing with the team building and the creation of new relationships should be addressed as these new teams are being formed. As the technical staff returns from weeks of training and begins to transition to the new development tools, dealing with the resistance will smooth the transition process. It is critical to anticipate the potential people issues that will arise with each step of the implementation and then act accordingly.

9. MONITOR THE CHANGE

As with any process improvement initiative, the change project must be monitored and adjusted. Since organizations are not static and technology is changing faster than we can anticipate, the actual change project may itself change during the

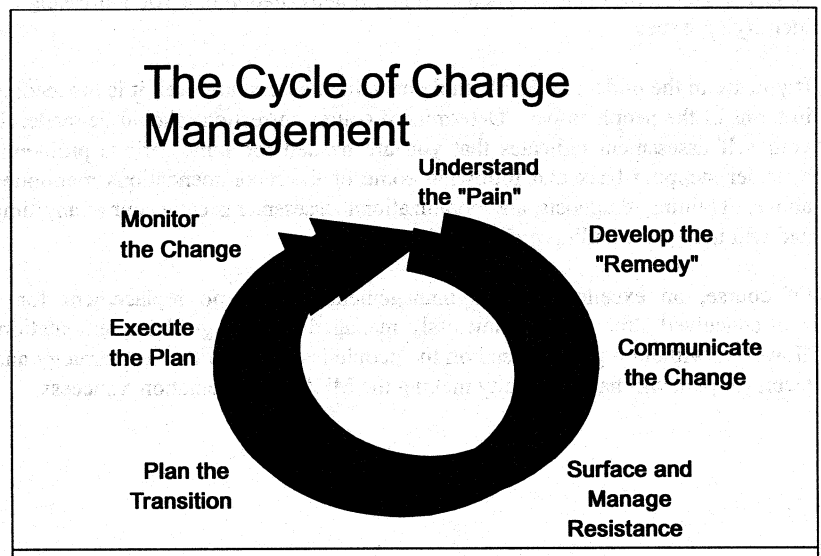


Illustration 3

MFA implementation. Key sponsors may change jobs, organizations may be merged or eliminated, new competitive pressures in the market may alter the direction of the project. All of these changes can affect how the technical implementation will proceed. With any major shift in the key variables, the change project will likely need to be re-evaluated. Monitoring these variables and making the necessary adjustments will enable the organization to keep the people transitions in synch with the technological change. (See Illustration 3.)

RECOMMENDATIONS

The methodology that has been described has identified a series of steps for managing the people issues associated with a major technology implementation such as a Mainframe Alternative project. However, these steps apply to any major change, be it organizational, process re-engineering, job restructuring or any other technological shift.

If you are just beginning to consider making a Mainframe Alternative decision, identify the people issues now and begin to put a plan in place. Train your management staff on change management methodologies so that everyone can speak the same language and understand the critical success factors for change. Diagnose how successful your change projects have been in the past, and learn from your past mistakes. Determine the level of impact that this technological shift will have on your employees and the extent to which the existing culture will pose a barrier to the change. Then craft an implementation plan for addressing the identifying issues.

If you are in the midst of a MFA implementation, assess how well it is proceeding in terms of the people issues. Determine if course corrections should be made. If your self assessment indicates that you are headed for some serious problems, consider stepping back and following some of the recommendations mentioned above. Training, diagnosis, and organizational assessments can occur at any time and will help put a flailing project back on track.

Of course, an excellent change management plan is no replacement for a well-conceived strategy or a flawlessly managed technological implementation. However, without a plan focused on the people issues, even the best strategy and technical plan will have difficulty making the MFA implementation a success.

HP Predictive Support File Transfer : Case Studies in Implementation

Paper #6021

Jim Rice

Hewlett-Packard Company
Response Center Lab
100 Mayfield Avenue
Mountain View, California 94043
(415) 691-3427

Abstract

This paper presents case studies from the HP Predictive Support product, which transfers files between HP3000/S900 machines at HP Response Centers around the world, and a large installed base of HP3000 Classic, HP3000/S900 and HP9000/S800 machines at customer sites. The case studies examine the technical options and tradeoffs considered in a multi-platform environment where security, supportability and wide availability are very important.

The paper begins with a brief survey of the following products and technologies, which will be referred to in the case studies:

File Transfer	DSCOPY; FTP; UUCP; and kermit
Modem Dialing	UUCP; and cu
Network Connectivity	NetIPC; Berkeley Sockets; SLIP; Email Server; HP DTC and DDFA Utilities

Survey of Products and Technologies

File Transfer

DSCOPY

DSCOPY allows HP3000 and HP9000 users to move files between machines. Users must either establish a logon to the remote machine (for example, using DSLINE between HP3000's) or provide a valid user and password (for example, when moving files from an HP9000 to an HP3000). DSCOPY is included in the NFT (Network File Transfer) component of HP's NS (Network Services) product.

FTP

File Transfer Protocol (FTP) is part of the ARPA/Berkeley Services protocol suite. It is designed to be a general purpose file transfer protocol that can be used on a variety of machines. Both the HP3000 MPE/iX and HP9000 HP-UX systems provide ftp programs, which use the protocol to transfer files between machines. An ftp user connects to a remote machine using an IP address or node name, and then specifies a user name and password. The ftp user may transfer files between the machines, browse files on the remote machine, and rename or delete remote files. An HP9000 user may also create or delete remote directories. On MPE/iX and HP-UX, the FTP product is shipped with the ARPA/Berkeley Services products.

On HP9000's, a system manager can configure *anonymous ftp* for extra security. A special ftp user is created which can only be used for anonymous ftp, not to start up a normal shell for a remote user. Once an ftp connection is established, ftp does a *chroot* on the destination machine so that the remote user cannot change to any directory outside the anonymous ftp directory - typically, */users/ftp* - and its subdirectories. Thus, a system manager can allow users to deposit files on and retrieve files from this machine without providing a normal user logon and password, or allowing access to files outside the anonymous ftp directories.

UUCP

UUCP (derived from the term "UNIX-to-UNIX copy") is a subsystem which can be used to transfer files between UNIX® systems and, to a limited degree, execute commands on remote systems. (UNIX is a registered trademark of AT&T in the U.S.A. and other countries.) UUCP is most useful when the machines are linked by RS232-C modem or direct connections, because it has flexible, table-driven modem dialing facilities. It provides batch transfer of files based on scheduled jobs, or interactive transfer. It allows the user to transfer files directly between two machines or through intermediate machines. Popular application programs that use UUCP include mailx, mail and similar mail handlers; notes; and news. UUCP is shipped standard with HP-UX.

Kermit

Kermit is a general-purpose terminal emulator and file transfer package, originally developed at Columbia University and ported to many systems, including HP-UX. Kermit allows a user to dial a modem, connect to, and log into a remote system. The user may then work on the remote system as if connected through a terminal. To transfer files, the user starts kermit on the remote machine and places both kermit processes (local and remote) in a special mode to transfer files. The user may then revert to the terminal emulator mode and continue normal work. Kermit is shipped standard with HP-UX.

Modem Dialing

UUCP

UUCP has flexible, table-driven modem dialing facilities. It uses configuration files to specify the type of modem on any given port, define the dialing sequence for a given modem type, and associate remote systems with ports and time frames during which batch transfers can take place. Users may easily write new dialing sequences for new modems.

Cu

Cu is a table-driven modem dialing and terminal emulator package for UNIX® machines. It uses a subset of the UUCP configuration files to specify the type

of modem on any port, and define dialing sequences. Cu is shipped standard with HP-UX.

Network Connectivity

NetIPC

NetIPC (Network InterProcess Communication) is a set of programmatic calls that allows applications on HP3000 MPE V/E, HP3000 MPE/iX or HP9000 HP-UX machines to exchange data between processes on different nodes in an NS network. These calls provide access to the transport layer, thereby allowing reliable communication. NetIPC is shipped with the NS Link product on HP3000 machines, and with the LAN product on HP9000/S800 machines.

Berkeley Sockets

Berkeley Sockets (or technically, the Berkeley Software Distribution Interprocess Communication facilities) offer functionality very similar to that of NetIPC. They are also a set of programmatic calls that allow applications on a wide variety of machines, including HP3000 MPE/iX and HP9000 HP-UX machines, to exchange data between processes on different nodes on a network. The type of Berkeley Sockets which are considered in this paper are stream sockets, which provide access to the transport layer, thus allowing reliable communication. It is assumed the network supports the Internet TCP (Transmission Control Protocol). Berkeley Sockets are shipped with the HP3000 NS Link product on MPE/iX, and with the HP9000 LAN product on HP-UX.

SLIP

SLIP (Serial Line Internet Protocol) is a protocol which allows TCP/IP applications to run over a serial link - for example, a modem connection. HP-UX implements SLIP with a feature called Point-to-Point Link (PPL), which is shipped with the LAN/9000 product. PPL can be run to dial into a remote machine, log in, and run PPL on that machine to set up a SLIP connection. Alternately, PPL can be configured on the remote machine to await callins and immediately set up a SLIP connection, thereby avoiding the need to provide a login to SLIP users. Once the SLIP connection is set up,

users can run TCP/IP applications (like ftp) to the remote machine by using the IP addresses or node names configured into PPL.

Email Server

Sendmail is the ARPA/Berkeley Services internetwork mail routing facility. It is used on HP-UX to route mail between mail agents and remote machines. An *email server* is set up by configuring a sendmail command line alias, which takes mail for a given address and pipes the message as standard input to a specified command, which is in effect the server. The client is the application or user which uses a mail agent, like *mailx*, to mail the message. Typically, the subject of the message indicates the operation the server should perform, and the body of the message contains the data needed for the operation.

HP DTC and DDFA Utilities

HP DTC's (Distributed Terminal Controllers) are general-purpose terminal servers which provide terminal and modem access to HP3000/S900 and HP9000 users, offloading much of this processing from the SPU. The HP-UX DDFA (DTC Device File Access) utilities allow HP-UX applications to access a modem, or a pool of modems, on a DTC located on the customer network as though the modems were attached to a MUX on the customer system. Only minor changes are necessary to most applications. The DDFA utilities are shipped standard with HP-UX as of 9.0.

The DDFA utilities provide access from HP-UX commands, like cu and kermit, or user applications to IP addressable HP DTC's through specially built device files which simulate those for local MUX ports. A DDFA utility (/etc/dpp) is run at system startup to read a configuration file (typically, /etc/ddfa/dp) and create special daemons - *ocd* daemons, or Outbound Connection Daemons - one for each DTC port configured for access. Each ocd daemon builds a special device file in /dev and waits for an application to access it. When an application opens one of these device files (for example, when cu dials the modem), the corresponding ocd daemon establishes a telnet connection from the system to the associated DTC port and provides pass-through access which is very similar to that for local MUX access.

The DTC manages most of the modem control. It asserts DTR, requires the modem signals to be high to accept an outbound connection, sets the baud rate to that configured for the port, passes data between the telnet connection and

the modem, and signals termination - by dropping DTR to the modem if the telnet connection is terminated, or by terminating the telnet connection if the modem control signals drop.

The DTC also provides modem pool functionality, whereby the DTC dynamically allocates free ports from a pool as callout requests are received from HP-UX machines. Normally, /etc/ddfa/dp is configured to associate a device file with the IP address of a DTC, plus a board and port on the DTC. However, if a modem pool is desired, the DTC is first configured to assign a common IP address to several modem ports. Then, /etc/ddfa/dp is configured with a device file and the IP address of the modem pool. Of course, multiple systems can be configured to share a single modem, or a pool of modems. By configuring a pool of modems and sharing it among systems, a customer can greatly increase the probability that users or applications will find a free modem.

The DTC is configured, downloaded and otherwise managed from either an HP9000/S800 or an HP Vectra PC on the same LAN as the DTC. A typical solution for a customer with only HP9000's would be to run HP DTC Manager/UX on an S800. A more convenient solution for a customer with both HP3000/S900's and HP9000's would probably be to share a DTC among these machines and manage it with HP Openview DTC Manager running on an HP Vectra.

Introduction to Predictive Datacomm

Predictive Overview

Predictive Support is a proactive diagnostic system which is provided to HP3000 Classic, HP3000/S900 and HP9000/S800 customers when they purchase a hardware support contract. (On the HP9000/S800 systems, it is shipped as part of the Support Watch product.) It runs nightly on the customer system, examines system logs and device-resident failure logs, and uses rule-based technology to detect potential hardware or memory failures. It notifies the HP Response Center via modem or network connection of any events which might require action. A Response Center Engineer (RCE) then diagnoses the problem, scheduling a service call by a Customer Engineer (CE)

to the customer system if necessary. Predictive Support helps to increase system availability by replacing unscheduled downtime with scheduled maintenance.

Predictive Datacomm Overview

When Predictive needs to notify the Response Center of a problem, it automatically calls out of the customer system, either by dialing a configured modem - usually the Remote Support Modem, provided by HP - or connecting via a network - for example, an X.25 connection over a leased line. It transfers the results of its analysis as a file to the Response Center machine, where an RCE is alerted to further analyze the problem.

Because Predictive's analysis of the customer system is rule-driven, new rules can be downloaded to the customer system as HP device experts refine them, without distributing a new release of Predictive. These rule updates are downloaded as files from the Response Center machine to the customer machine. (The term *download* will always be used in this paper to refer to a transfer from HP to the customer system; an *upload* goes from the customer to HP.) For security reasons, rules are only downloaded if the customer system has initiated the connection to HP; the Response Center machine never automatically dials in or connects to the customer machine.

Predictive uses a codeword for software protection. During installation and configuration, the serial and model number as entered by the CE are transferred to the Response Center in a file. The phone line or network connection is held open for up to five minutes while software at the Response Center checks for proper contract coverage, creates a codeword and downloads it in a file. The codeword is encrypted in such a way that it can only be used on that machine. It allows Predictive to be scheduled for nightly analysis.

Technical Requirements for Predictive Datacomm

The following requirements are of *HIGH* importance.

- Security
  - Predictive must always initiate connections from the customer system to the Response Center.

- Predictive can never automatically log in to the Response Center machine. This restriction is part of HP Corporate Security Standards to protect sensitive customer data.

- Supportability
  - Preferably, the Predictive datacomm software will use components which are standard HP supported products. If software is adapted from outside HP, it should adhere to de facto industry standards.
- Wide Availability
  - Because Predictive is included with standard hardware contracts, it cannot require the customer to have any software which is not included as part of the standard system. If Predictive offers an option - for example, network transfer - it may require software that is not standard, but should ensure this software is widely available and commonly installed.

The following requirements are of *MEDIUM* importance.

- Flexibility
  - As new HP Remote Support Modems are introduced, Predictive datacomm must support them.
  - Predictive datacomm should adapt to new transmission mediums as they become generally available - for example, network connections.
- Ease of Installation
  - It should be easy to install and configure Predictive datacomm.

The following requirement is of *LOW* importance.

- Performance
  - Predictive needs to transfer the results of nightly runs infrequently, and these are typically small files. Ruleset updates are larger (for example, 50 KBytes), but are only downloaded to customers once or twice a year. As the Predictive product evolves, it may well become more important to transer large amounts of data quickly.

Case Study 1 : HP3000 to HP3000 using Modems

This case study discusses the architecture used to meet the basic technical requirements for customers running HP3000 machines and transferring to HP3000 machines at the Response Centers. This architecture serves equally well for Classic HP3000's and HP3000/S900's at the customer site, or at the Response Center. It has been in use since the earliest releases of Predictive.

The following diagram illustrates Predictive connections to the Response Center from three customer systems : an HP3000 Classic, an HP3000/S900 connecting through an HP DTC, and an HP3000/S900 connecting through the remote console port. (The latter configuration is possible on more recent S900's like the 9x7 and 99x machines.) All systems dial into the same bank of modems at the local Response Center.

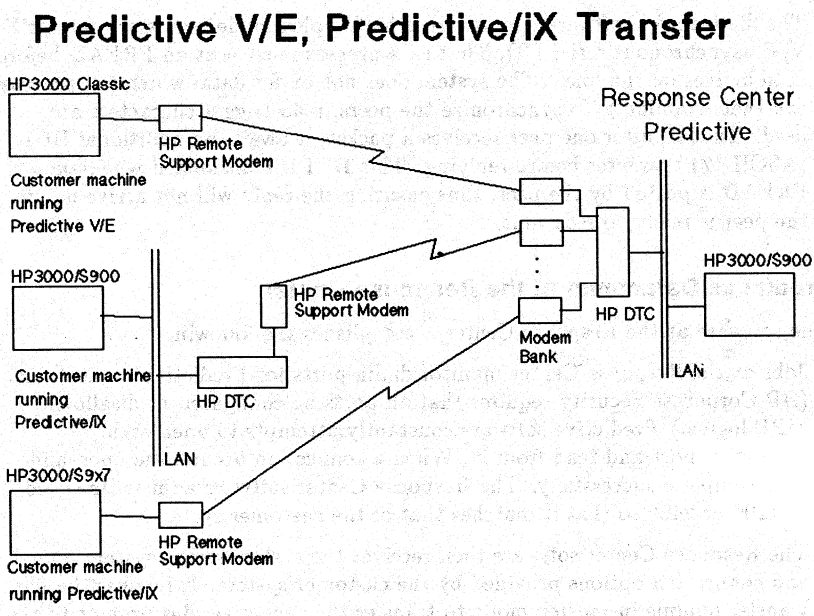


Figure 6021-1.

Predictive Datacomm on the Customer Machine

The software on the customer system accomplishes the following:

- Modem dialing software interrogates the modem to determine its type and dial it. All HP Remote Support modems can be dialed.
- Upon connection, Predictive transmits the system serial and model numbers plus options indicating the type of transfer requested. It then reverts to a *slave* mode, in which the Response Center software indicates files to transfer to or from the Response Center.
- File transfer software was written especially for Predictive. This software can transfer any standard MPE file type, binary or ASCII, including user file labels. A file is transferred as a series of packets, each of which is typically the size of a file record - for example, 128 words for binary file. Packets are checksummed and retransmitted up to 10 times on error.

  The file transfer software operates in a half-duplex mode, necessary for MPE V/E asynchronous serial I/O. That is, a process must post an FREAD before data arrives on the line. (The system does not buffer data, which the process can then examine.) To synchronize the peers, read trigger characters are used. That is, after one peer receives a packet, it awaits an additional DC1 (ASCII 17) character before replying. This DC1 is transmitted when an FREAD is posted by the peer, thus ensuring the reply will not arrive before the peer is ready to receive it.

Predictive Datacomm at the Response Center

The software at the Response Center accomplishes the following:

- Jobs at the Response Center monitor dialin ports for Predictive connections. (HP Corporate Security requires that all ports be configured to disallow MPE logins.) Predictive software constantly attempts to open each Predictive port and read from it. When a connection occurs, the open and read complete successfully. The Response Center software adjusts its speed to 1200 or 2400 so that it matches that of the customer system.
- The Response Center software then receives the system serial, system model, and connection options provided by the customer system. It invokes the file transfer module in *master* mode to transfer the necessary files from or to the

customer system. This is the same file transfer module used on the customer system.

- The Response Center software has access to a database of scheduled downloads, indexed by system serial and model number. These are typically ruleset download files or codeword files.

Discussion

This scheme meets all of the basic requirements. A few points are noteworthy:

- Security is maintained because no logins are used on either the customer or Response Centers machines. All connections are initiated from the customer machine, and the Response Center jobs monitor the modem ports, which are always configured to prevent MPE logins.
- No special software is required - only that which is shipped standard with MPE and Predictive.

Case Study 2 : HP9000 to HP3000 using Modems

This case study discusses extensions to support customers running Predictive/UX as part of Support Watch on HP9000/S800 machines. These machines connect to an HP3000/S900 machine at the Response Center.

The options considered when porting Predictive datacomm to HP-UX included:

- use *UUCP* to dial the modem and transfer files to and from the Response Center
- use *PPL* to dial the modem and and establish a *SLIP* connection to the Response Center; use ftp to transfer files to and from the Response Center
- use *cu* to dial the modem and *kermit* to transfer files to and from the Response Center

UUCP

The extensions to Predictive/UX would involve using UUCP to dial the modem and transfer files to a front-end HP9000 at the Response Center (either a workstation or an S800). From here, a daemon process would forward the files to the HP3000 where the rest of the Response Center Predictive software exists. Files to be downloaded would be placed on the front-end HP9000. Predictive/UX would use UUCP to dial in a short time later and retrieve any such downloads.

UUCP offers the following *benefits*:

- It would allow Predictive/UX to relay files from one customer machine through an intermediate machine. This *hub transfer* (where the intermediate machine is referred to as a hub) allows flexibility for customers who do not wish to attach a modem to every machine. For example, if configured properly, Predictive/UX on machine *satellite1* could transfer through *hub1* to *HPRC1* with the following UUCP command:

```
uucp predfile hub1!HPRC1/usr/spool/uucppublic/predfile
```

- Software is available which allows UUCP to operate over a *network* link. When configured correctly, UUCP's uucico program can use the *vt* utility on the local system and the *vtydaemon* to transfer over a LAN connection instead of over a serial connection. This would allow connections from a satellite over a LAN to a hub, and then from the hub over a modem to the Response Center.

However, the UUCP solution had the following *drawbacks*:

- The configuration of UUCP can be difficult.
- This option would require the purchase, installation and maintenance of an HP9000 front-end machine at each Response Center worldwide.
- New software would need to be written to relay files between the front-end HP9000 and the HP3000 running Response Center Predictive.

SLIP and ftp

As with the UUCP option, this option would involve using an HP9000 as a front-end at the Response Center. PPL would be run continuously on this machine, waiting for callins to device files dedicated to SLIP. Whenever such a callin occurred, PPL would establish a SLIP connection immediately on the line. On the customer machine, PPL would be configured to dial the modem using the UUCP table-driven dialing software, connect to the HP9000 at the Response Center and establish an SLIP layer.

Then ftp would be run over this link to transfer files to the HP9000. Downloads would be placed on the HP9000 at the Response Center, and the customer machine, under the control of Predictive/UX, would keep checking until any download files appeared, or a predefined period of time expired.

This option offers the following *benefits*:

- It would allow Predictive/UX to relay files from one customer machine through an intermediate machine. Predictive/UX could use PPL on a hub to establish a SLIP connection to the Response Center. Then, with the proper IP routing table entries, ftp could be run on another customer machine to connect through the hub to the front-end machine at the Response Center.
- This option is easily extended to a network link. That is, ftp could be run directly over a network link instead of SLIP.

However, this option had the following *drawbacks*:

- At the time Predictive was ported to HP-UX, the LAN networking product, which includes PPL, was not standard or purchased by all S800 customers.
- This option would require the purchase, installation and maintenance of an HP9000 front-end machine at each Response Center worldwide.
- New software would need to be written to relay files between the front-end HP9000 and the HP3000 running Response Center Predictive.
- This option is less secure than others because SLIP can provide a dialin user with general access if not configured properly. That is, the machine which SLIP runs on should be configured to accept only ftp connections.

Cu and Kermit

This option involves developing software on the HP9000/S800 to launch cu to dial the modem, then launch kermit to transfer files. Kermit is run under cu because allowing cu to terminate would cause it to hang up the modem. Cu is instructed to stop its receiving process when launching kermit (with a ~&kermit command) to prevent it from interfering with kermit's reads from the port.

A version of kermit for the HP3000 is used on the Response Center machine. (This version was contributed to the Interex Contributed Library by Telamon Systems.) To be compatible with this version, kermit on the S800 is run in half-duplex mode, using a DC1 read trigger and 90-byte packets.

The Response Center jobs receive Predictive connections from both HP3000 and HP9000 systems on the same phone lines, and determine from the initial packet of information sent over the line which type of system has connected. For Predictive/UX connections, they invoke kermit for file transfers.

This option offers the following *benefits*:

- Cu and kermit are easy to configure. Simple scripts can be written to set up the two configuration files required for callout : */usr/lib/uucp/Systems* and */usr/lib/uucp/Dialers*.
- This option required no new hardware at the Response Center. In fact, it required no new jobs because the same Response Center job accepts Predictive/iX and Predictive/UX connections.

This option has the following *drawbacks*:

- It does not offer *hub transfer*, that is, the ability to route through an intermediate machine.
- It is not expandable to transfer over a *network* link.

Discussion

The option to use cu and kermit was chosen for Predictive/UX. The deciding factors were the ability to use the same Response Center hardware and jobs for all types of Predictive connections, and the standard availability of cu and kermit.

However, both UUCP and SLIP can be attractive options if it is not required to integrate transparently with HP3000's. SLIP is especially attractive because it allows TCP/IP applications transparent access over modem and network connections. It can be configured to log in to a remote machine and then set up the IP layer, or the remote machine can be configured to have SLIP servers running on the modem ports. If SLIP connections are set up without login, the inetd daemon should probably be configured to regulate the applications which can be executed.

Case Study 3 : HP3000 to HP3000 using Network Connections

This case study discusses extensions to support customers running Predictive/iX and wishing to transfer over a network connection to the Response Center. This functionality was originally developed for customers who connect using X.25, but is extensible to TCP/IP network connections.

The options considered when enhancing Predictive datacomm to support network connections included:

- use NS services - that is, *DSLINE* and *DSCOPY* - to establish a network connection and transfer files
- use *FTP* to establish a network connection and transfer files
- use *Berkeley sockets* to establish a client/server connection, and the existing Predictive protocol to transfer files
- use *NetIPC* to establish a client/server connection, and the existing Predictive protocol to transfer files

DSCOPY

In this design, Predictive/iX would have opened a DSLINE from the customer system to the Response Center machine and logged in. It then would have transferred files to the Response Center using DSCOPY, waited for any downloads to be placed in a holding group and account, and transferred these to the customer machine.

This option had the following *drawback*:

- This option violates HP Corporate Security Standards by requiring a logon to the Response Center machine.

FTP

In this design, Predictive/iX would have invoked FTP on the customer system to establish a connection to the Response Center machine. It then would have transferred files to the Response Center, waited for any downloads to be placed in a holding group and account, and transferred these to the customer machine.

This option had the following *drawback*:

- This option also violates HP Corporate Security Standards by requiring a logon to the Response Center machine.

Berkeley Sockets

In this design, Predictive/iX would use Berkeley stream (or TCP) sockets to establish a client/server relationship between a Predictive/iX process on the customer machine and a process on the Response Center machine. The Response Center Predictive server would create a socket on a "well-known TCP port" and the Predictive/iX client would connect to it. Once the connection was established, Predictive/iX would use much the same protocol it does to transfer files over a modem. The main difference would be that Berkeley socket calls would replace FREAD's, FWRITE's and FCONTROL's to a modem port. Furthermore, there is no need for Predictive/iX to checksum packets or transfer acknowledgements, because this is handled by the transport layer when stream (TCP) sockets are used. Predictive/iX must check for errors on calls to Berkeley sockets, in case the connection to the peer is lost.

In order to establish a connection between a client using Berkeley Sockets and a server using NetIPC, the following calls are used:

Table 6021-1. Berkeley Sockets to NetIPC Connection

Phase	Berkeley Sockets Client	NetIPC Server
Interface Setup	socket() - create socket; *gethostbyname(), *getserverbyname() - create address structure for server's IP address and TCP protocol relative address	ipccreate() - create socket at well-known TCP protocol relative address; Berkeley Sockets client cannot access NS Socket Registry used by ipcname()
Connection Establishment	connect() - request connection to server, block until server has accepted	ipcrecvn() - accept connection request, block until receipt
Data Transfer	send()/recv(), or write()/read()	ipcsend()/ipcrecv()
Connection Release	close()	ipcshutdown()

However, this option had the following *drawbacks*:

- When this Predictive functionality was added, Berkeley sockets was not shipped standard with the NS Link product, and thus was not available on all customer machines with network connections.

NetIPC

This design is very similar to that involving Berkeley sockets. Here, a Dispatcher process is used on the Response Center machine to accept connections and launch a child Controller process for each Predictive connect. This allows concurrent Predictive connections on the same call socket, but different virtual circuit sockets.

Table 6021-2. Sample Predictive/IX Connection using NetIPC

Customer System	Response Center
	Dispatcher process creates call socket (IPCCREATE)
	Dispatcher names socket using well-known NS socket (IPCNAME)
	Dispatcher awaits connection (IPCRECVN)
Create call socket (IPCCREATE)	*blocked, awaiting connect*
Search socket registry for well-known socket and RC Predictive node name (IPCLOOKUP)	*blocked, awaiting connect*
Connect to well-known socket, creating VC (Virtual Circuit) (IPCCONNECT)	*VC socket returned*
Check status of connection (IPCRECV)	*Peers can communicate using VC sockets*
	Dispatcher releases VC socket (IPCGIVE)
	Dispatcher creates child process Controller; Dispatcher continues, to await next connection
	Controller process starts and receives VC socket (IPCGET)
System Identification (IPCSEND)	Receive Identification (IPCRECV)
Connection Options (IPCSEND)	Receive Options (IPCRECV)
Receive next command (IPCRECV)	Send next command : upload (IPCSEND)
Send File (IPCSEND)	Receive File (IPCRECV)
send record by record	*receive record by record*
Receive next command (IPCRECV)	Send next command : disconnect (IPCSEND)
Shut down connection (IPCSHUTDOWN)	Shut down connection (IPCSHUTDOWN)

Discussion

The NetIPC option was chosen for Predictive/iX. Both the DSCOPY and FTP options were rejected because they required logons to a Response Center machine. FTP does not require the user to establish an MPE session on the remote machine, but it does require the user to provide a valid user and password. Even if this were encoded in the software, its use would still constitute a risk to HP Internet security.

The NetIPC option was chosen because the Berkeley sockets calls were not included in the NS3000/XL Link product at the time. However, in more recent releases of NS3000/XL Link, Berkeley sockets is included. Both are excellent solutions for MPE/iX applications. Berkeley sockets is recommended for long-term support for applications which will run on HP-UX.

Case Study 4 : HP9000 to HP3000 using a Combination of Network and Modem Connections

This case study discusses extensions to support customers running Predictive on HP-UX machines and wishing to use their private network to minimize or eliminate the use of modems. This functionality was initially designed for a customer with a large number of S800's, most of which are unattended and connected by a private X.25 network.

The options considered included:

- extend the Predictive/iX network solution to HP-UX, using *Berkeley sockets* and a *client/server* approach to eliminate the use of modems
- provide a *store-and-forward* solution using *ftp* to a central machine, then the standard Predictive/UX modem connection to the Response Center
- provide a *store-and-forward* solution using *sendmail* to a central machine, then the standard Predictive/UX modem connection to the Response Center
- install an *HP DTC (Distributed Terminal Controller)* at a central customer site to provide *network modem* capability; use the *HP-UX DDFA (DTC Device File Access)* utilities and the standard Predictive/UX modem software to establish connections to the Response Center

Client/Server with Berkeley Sockets

This option would have extended the Predictive/iX network solution to Predictive/UX, using Berkeley sockets in place of NetIPC. We would also have to extend the file transfer protocol to handle HP-UX regular files as well as MPE files. We would then be able to have a single Predictive server on the Response Center machine using NetIPC to accept connections from Predictive/iX clients using NetIPC or from Predictive/UX clients using Berkeley sockets.

This option offers the following *benefits*:

- It eliminates the use of modems entirely.

However, this option had the following *drawback*:

- The customer (who is based in the United Kingdom) wished to connect to the local Response Center using X.25; because of separate security concerns, the customer did not wish to connect through a PDN (Public Data Network), and HP wanted to avoid the installation of a new X.25 leased line into the local Response Center.

Store-and-forward using ftp and Predictive/UX Modem Transfer

This option would have used anonymous ftp to move files between satellite machines and a central customer machine, from which a modem connection could be made to the Response Center. (Satellites have network connections to a central machine; the central machine has a modem connection to the HP Response Center.) A Predictive/UX process on the central machine would constantly poll for files in an anonymous ftp directory created for Predictive/UX, then invoke Predictive/UX datacomm to transfer these to the Response Center, and download any files scheduled for the given satellite.

This option offers the following *benefits*:

- Ftp is simple to use.

However, this option had the following *drawbacks*:

- It requires the customer to configure the central machine for anonymous ftp. This creates files outside the Predictive/UX directory, if only temporarily.

- It is difficult to provide real-time status to a user on the satellite during a connection - for example, while a modem is dialed.

Store-and-forward using sendmail and Predictive/UX Modem Transfer

This option uses a store-and-forward approach, similar to the previous option. It sets up Predictive/UX *email servers* on the central machine and on the satellites to route files and request connections.

The following steps could occur in a "connection":

- Mailx would be used to transfer files from the satellite to the central machine. First, the files would be packaged using *shar*.
- The Predictive/UX email server on the central machine would receive the Predictive/UX files, unpackage the shar file, move the files to a holding directory, and invoke Predictive/UX modem transfer to connect to the Response Center on behalf of the satellite.
- Any files downloaded for the satellite would be sent back to the satellite using mailx. They would first be packaged with shar.
- The Predictive/UX email server on the satellite would receive the files, unshar them, move the downloaded files to the Predictive/UX directory and inform Predictive/UX of their arrival.
- Finally, success or error status could be emailed from the central machine to the satellite email server.

This option had the following *benefits*:

- Sendmail provides a simple interface.

However, this option had the following *drawbacks*:

- It is difficult to provide real-time status to a user on the satellite during a connection - for example, while a modem is dialed.
- If sendmail is not configured to correctly route to the central machine, error handling is not elegant. That is, a message which is sent to the central machine but does not arrive is probably "bounced" and placed in the root mailbox on some system, not necessarily the initiating client. The system manager may not know how to troubleshoot the problem. Furthermore,

Predictive/UX on the client probably has no easy way to determine where the problem lies - it simply detects that no message is returned from the central machine to indicate success or failure.

Network Modem using HP DTC and DDFA Utilities

In this option, the customer machine is configured to use the DDFA utilities to access a modem, or a pool of modems, on a DTC on the customer network. Then, Predictive/UX modem transfer software is run as normal. The following shows a typical hardware configuration for a customer running Predictive/UX on many systems, sharing a modem pool on a DTC:

Predictive/UX Transfer via HP DTC

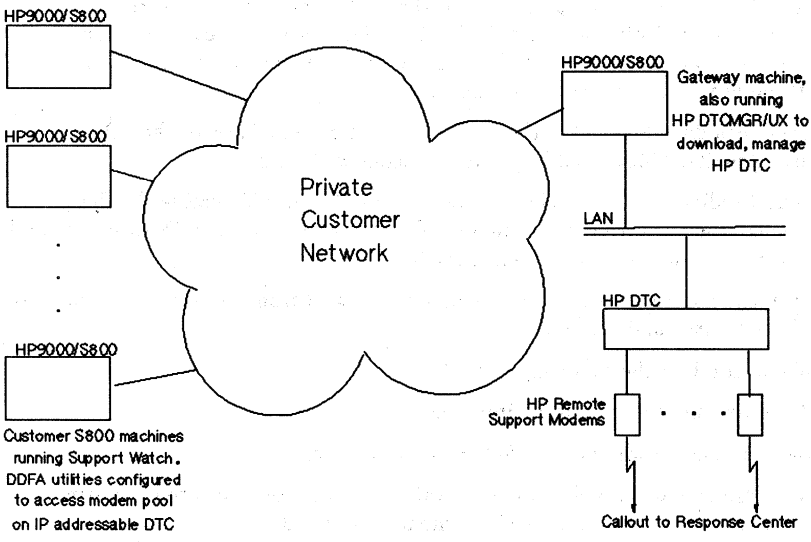


Figure 6021-2.

Minor changes were made to the the modem transfer software due to limitations of the DDFA utilities:

- Parity cannot be changed dynamically. It must be configured for the port through the DTC management software, and thus cannot be changed during a connection. Predictive/UX and Response Center Predictive were both modified to carry out connections without parity.
- Baud rate cannot be changed easily on a port. It is also configured for the port through the DTC management software. Whereas Predictive/UX normally drops the modem speed from 2400 baud to 1200 baud on connection retries, the modified Predictive/UX always dials at 2400 baud.

Furthermore, because of the latency which can be introduced by the network link - for example, a customer's private X.25 network - several waits and timeout intervals were increased or made configurable.

This option has the following *benefits*:

- Its development cost is the lowest of all options considered.
- It provides virtually the same behavior as normal Predictive/UX modem transfer - for example, real-time transfers (as opposed to store-and-forward) and very good error handling.

However, this option has the following *drawbacks*:

- The customer must buy, install and maintain a DTC on the customer network; however, when this is compared to the cost for some network connections - for example, an X.25 leased line to HP - the cost is acceptable.

Discussion

The DTC solution was chosen for this type of customer. It provides a low cost development solution. For any application which dials out from multiple HP-UX machines, where it is not convenient or cost-effective to establish network connections, this solution can reduce cost and allow more centralized, secure management than placing a modem and phone line on each system.

PAPER NUMBER: 6024

TITLE: PowerHouse: The Technology Behind the 4GL

PRESENTER: Bob Gibb
Cognos Corporation
3755 Riverside Drive
Ottawa, Ontario K1G 3Z4

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Why and How You Should Move to SQL

David Wiseman

Proactive Systems

Introduction

A year ago I presented a paper to a users conference entitled "Can I Use SQL?". That paper compared today's IT departments to Christopher Columbus setting out across the uncharted waters of future technological development, and suggested that, as far as database development was concerned the only reliable compass available was the SQL standard. Since then, technical and commercial developments have combined to present a clearer picture. We are now rather like Columbus sighting the distant shores of the Americas: we can see land, but it is not yet clear exactly what we will have to confront when we get there. In this paper, I will try to show why you not only can, but should use SQL for new applications development, in the light of the recent evolution of the market. I will also endeavour to provide some rough charts of the new Americas which will go a bit further than the traditional "Here be monsters". Since the "Old World" that we are all starting out from is essentially the HP3000 and IMAGE environment, I will be examining in particular the implications of moving to SQL from IMAGE.

What is SQL?

Before we enter our subject proper, it is worth just briefly considering what SQL is, since it has existed long enough as a buzz-word to spread considerable confusion. It is important to be aware that "SQL" is not, in itself, a database or even a type of database. SQL is a database access language which provides programmers and applications packages with a standard means of accessing data. Although SQL is appropriate to relational methodology, and has indeed grown up with relational databases, the language does not in itself make any conditions about the underlying file structure. This of course is one reason that SQL works as a standard: it is not tied to any particular operating system, file type, etc. In principle, any database that can satisfy an SQL request will work with any software that can issue such a request. This fact has led some commentators to say that the introduction of SQL for IMAGE has given HP, at a stroke, the world's largest installed base relational database! Generally speaking, however, the use of SQL implies a much wider change towards the use of true relational databases, which will often be accompanied by the implementation of client-server technology and Graphical User Interfaces (GUIs).

It is important to be aware that SQL is itself divided into two subsets: the DML (Data Management Language) commands which carry out all the functions that we normally associate with programming (add, modify, and delete data); and the DDL (Data Definition Language) commands which allow the Data Base Administrator to define the database (tables, file spaces, indexes, etc), in other words to execute the functions which in IMAGE would be carried out using DBSCHEMA and DBUTIL. This can have quite profound implications; for example, the DBA can build a new index on a table simply by executing the CREATE INDEX command, which is a good deal easier than the equivalent process in IMAGE (add an automatic master to the database, add the path between detail and master, etc etc). Moreover, it is possible to create indexes in this way without having exclusive access to the database.

Until quite recently, a major problem with the SQL (ANSI 89) standard was poor functionality. For RDBMS manufacturers such as Oracle to provide an adequate command language, they had to include a plethora of commands which

were not included in the SQL standard, and this in turn made it more difficult to mix and match products. Today, however, the arrival of the SQL2 standard offers very substantial improvements in functionality, including features such as triggers and procedures, for example, and the major RDBMS suppliers have followed this standard. In terms of ALLBASE/SQL, this corresponds to the move from ALLBASE "E" to ALLBASE "F" (under MPE/iX 4.0), and indeed ALLBASE now provides a very good implementation of the SQL2 Part 2 standard.

There is a major difference between programming in SQL and programming, for example with IMAGE. When you write a program in Cobol to access an IMAGE database, there is no inherent connection between the program and the database. The source code can be compiled and linked without the database even being present on the machine. This is not the case with SQL (irrespective of which RDBMS you are using; we will not, here, go into the subject of dynamic pre-processing, which is not really technically relevant to applications development). Programming with SQL introduces a third step, which has to be carried out before compilation: pre-processing the source code, to replace SQL statements with the appropriate calls to the procedures supplied with the RDBMS. The SQL statements are "compiled" and stored as modules in the database by the pre-processor, while the modifed source code can then be compiled and linked in the normal way. This means that any RDBMS will also include pre-processors for the various languages (Cobol, C etc) that it supports.

Why using SQL makes sense

It is a truism in computing that by far the largest cost in IT is not the hardware, but the applications software. This is all the more true if we add to the nominal cost of buying or developing an applications package, the hidden but nonetheless all too real costs of installing a new package, in terms of replacing old systems, user training, etc. Once an application has been installed and has entered into a company's "normal practice", it acquires an inertia of its own, simply because of the difficulty and cost of retraining users. Consequently, the decisions that an IT department confronts when preparing a major new development project are never purely technical: they are also, and perhaps even more importantly, economic. We have to be assured not only that the technical solution that we choose to carry out the project will work, but that it will continue to work for the next ten or even fifteen years, and equally important

that it will continue to be supported (we do not want to fall into the same trap as the users of Virtuoso). Moreover, it must have the potential to take advantage of new technology, and to interface with other applications which may or may not run on the same platform. It will have to adapt to change, either in technology or in business practice, much more quickly than before.

Looked at in this light, it is clear that there is at the very least a serious question-mark hanging over IMAGE as a suitable tool for undertaking major new developments, and this for reasons which are as much economic as technical. For one thing, IMAGE will obviously never be ported onto other platforms than the HP3000, so that the portability of applications written in IMAGE will be nil. It will also limit severely the choice of third-party applications packages to those designed solely for the HP3000 market, as well as the ability to use applications development packages, especially those designed for client-server environments. Nor is it even certain that, in ten years time, the HP3000 will still be around in the way we know it today. We may end up with MPE as a shell on top of a Posix-based operating system. All this suggests that, although we can be confident that HP will continue to support MPE and IMAGE for as long as customers need them, neither is likely to remain in the mainstream of technological development. And no company can afford to see its "mission-critical" applications stuck in a technological dead-end.

Let it be absolutely clear here that these arguments are not intended to "do down" IMAGE. Those of us who have worked with IMAGE have a great deal of respect for its reliability, and it has served its users extremely well over the years. However, the arguments that we have just put forward are not essentially technical, but economic, and as one ex-Prime Minister remarked "You can't buck the market". To this we might add that IMAGE was, after all, developed at a time when a 512 Kbyte memory upgrade on an HP3000 was a major event: today, when main memory is measured in tens of megabytes, and shortly in gigabytes, we should expect that database technology will advance to take account of developments in hardware. This in turn will have economic implications, since it is obviously not commercially sensible for a DBMS supplier to commit the same level of development effort to old and new-technology DBMS.

How can using SQL help to ensure the perennity of our applications?

1) By allowing the "transferability" of personnel. Today, knowing how to develop applications in IMAGE is a specialized skill, which new programmers have to learn from scratch and which cannot be transferred from one operating system to another (and it is worth remembering that most companies have at least two operating systems installed - DOS/Windows, and something else). SQL, on the other hand, is already a standard language taught in technical colleges and universities. Using SQL frees the IT department from dependence on specialized IMAGE skills, as well as making it possible to transfer personnel more easily between different platforms.

2) By facilitating hardware connectivity. If we use databases which are able to satisfy SQL calls, then programs running on different hardware platforms do not need to know anything about the host's file structure, etc, in order to request data from it. In fact if we use development tools such as Oracle's SQL*Net or Gupta's SQL APIs, then we do not even need to know the location of the host databases that we are going to access.

3) By facilitating software connectivity. If anything, this is even more important that hardware connectivity, and we can expect its importance to increase. Using SQL allows us to open up our databases to all kinds of applications packages or tools. Most top-range spreadsheets, for example, now offer the ability to import data from external data sources through embedded SQL commands.

4) By making programs independent of the underlying data structure. If in ten years time, we find that technical development has made present file systems outdated, we can change the underlying structure of the data without having to modify the programs that access it, since these only need to know how to request the data in SQL and need know nothing about the file system underneath.

5) By providing a path towards new technologies. At present, this means above all a path towards Object Oriented databases (OODBMS). If we take HP as an example, we can see first of all that ALLBASE already contains the beginnings of OO data structures (triggers, procedures, BLOBs), while the language being used to access Object data is OSQL, in other words an extension of the already existing standard SQL.

Now is a good time to start moving to SQL

The problem with moving towards new technologies always lies in knowing when to start. If we imagine technological evolution as an "envelope" of current technologies, then we can see first, that it will be constantly moving, and second that the envelope will contain both new technology at the front of the envelope (what some like to call the "cutting edge"), with old technology at the back. A bit like this, in fact:

..... dead .....(old . . . . . new) emerging --> >

The problem for an IT department is first to determine in which direction the envelope is moving, and then to invest in such a way that we always stay within the envelope of current technology, without either going out on a limb with something so new that it is not yet tried, proven, and workable, or dropping off the back and getting stuck with something so out of date that it is incredibly expensive to change. Since any investment we make is bound to possess inertia, then over time the situation will look like this:

Initial investment +

..... dead .....(old . . . . . new) emerging --> >

After time elapsed +

..... dead .....(old . . . . . new) emerging --> >

In this example, our initial investment (machines, application software etc, indicated by the "+") has of course remained static, while the envelope has moved past it.

Let us consider this model in relation to SQL, and more specifically to SQL in an HP3000 environment. About eight years ago, when ALLBASE/SQL first began to be talked about, it fell very definitely into the "emerging" category. ALLBASE under MPE/V ran like a dog and was never a feasible DBMS (just as the first release of Oracle on the HP3000 suffered from serious performance problems). Performance under MPE/XL was acceptable (just about), but since ALLBASE functionality adhered closely to the then ANSI standard, and this

standard was extremely limited, it was simply not in the running as a tool to replace tried and tested IMAGE.

In fact the situation looked rather like this:

Image Sql

..... dead .....(old . . . . . new) emerging -->>

Today, things are very different. Both standards and the market have changed radically, in a number of ways:

1) As we said above, the SQL2 standard now contains a very substantial set of features, and in fact is well ahead of the RDBMS that you can currently buy. This makes it possible for an RDBMS to provide good functionality while still adhering to the standard, which in turn makes it easier to interface different tools. In other words, SQL databases are becoming truly open. They also include features that IMAGE developers take for granted such as row-level locking, and others that open up new avenues for database management, such as stored procedures and triggers.

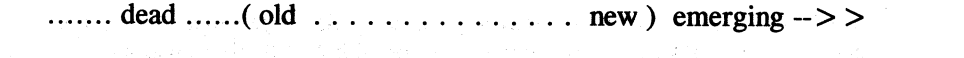
2) Performance has improved radically, as a result of developments in both hardware and software. In fact, our experience with ALLBASE suggests that we are now reaching a point where, if it is programmed correctly, SQL can outperform IMAGE in certain conditions (eg Select statements on large sets of data). Other manufacturers have also been working to improve their performance.

3) The tools available for developing under SQL are now numerous enough to provide a substantial choice, and well-established enough to protect your investment. Initially, developing under ALLBASE meant using either a 3GL like Cobol, or ALLBASE/4GL which was non-standard and had a negligible installed base. Now it is possible either to use one of several well-established RDBMS (Oracle, Informix, Sybase, Ingres), or to combine ALLBASE with well-accepted tools such as Gupta SQL, or the ALLBASE/SQL PC APIs which can give you access to PC development tools.

4) As the SQL standard has become more complete, naturally the structure and functionality of the different RDBMS are coming closer together. In fact, we may be approaching the point where the DBMS can be considered as a commodity product, or indeed as part of the file system itself. This implies a separation between database and development tools, and we have seen a whole series of development tools emerge which will work with any of a number of different RDBMS. Some of these are now quite well-established, and offer a proven development environment: we can cite Uniface and Powerhouse as examples.

All these developments have combined to change the envelope so that it now looks like this:

In ten years time, we can expect the picture to look like this:



There is a very important proviso to be added here. Although we often use the term "SQL database" to refer to new generation RDBMS, we should always remember that, as we said at the beginning of this paper, SQL is not a technology as such but a language. An "SQL database" is therefore more correctly defined as "a database capable of satisfying calls made in SQL". In this sense, we can say that although in ten years time today's RDBMS will be moving to the back of the technology envelope, there is no reason to suppose that the same will be true of the SQL language, especially since we can expect OSQL to be an enrichment, rather than a replacement, of SQL itself.

(Before going any further, I would like to add that I am indebted to Andy Greenawalt, of Air Products in the US, for this very useful "envelope" image).

The conclusion that we come to is this: new applications should be considered with a view to developing under some form of SQL, rather than under IMAGE.

This in turn raises two major questions, which we will try to deal with next. First, how do we develop under SQL, and second, how do we share data between our existing IMAGE applications and the new SQL ones?

How do we develop with SQL?

Let's be clear first of all that I do not propose here to make specific recommendations, but to indicate several different directions that you might follow. Clearly, the route that any company takes will be very much determined by its own circumstances and requirements.

I have divided the possible routes into three, each of which of course has its advantages and disadvantages:
- do-it-yourself
- happy families
- pick'n'mix

1) DIY SQL
The main advantage of DIY is that it is cheap, and this is the kind of solution that may be suitable if you just want to undertake a pilot application, without investing in expensive software, and without having to retrain your programmers to use a new development tool. In the HP world, the cheapest SQL DB that you can buy is certainly either

ALLBASE/SQL, or IMAGE/SQL on the HP3000 (more of this later), and you can program with these in Cobol and VPlus, for example, which you may very well have on your system already. This solution involves minimum expense, but on the other hand you may find that it is too limiting to make an adequate pilot: for example, you may want your pilot application to try out client server technology as well. All is not lost however, since you also have available APIs for the PC environment supplied by HP, which you could use from within a programming tool like Visual Basic on the PC, as well as ALLBASE support from the Gupta APIs. The main disadvantage of the DIY solution is that it may prove inadequate to tackle major applications, since it does not use the available development tool technology to cut down the time and therefore the costs of development.

2) Happy Families SQL
This is the complete opposite of the DIY solution, since it means buying a complete package of both RDBMS and development tools from a single supplier such as Oracle, Ingres, and the like. Although one should always be sceptical about claims to provide "hardware independence" (since the hidden cost often tends to be software dependence instead), there can be substantial advantages to be gained in choosing the "single-vendor" solution. These include ease of administration and unity in applications development, which may be especially important in a multi-vendor hardware environment. Points to look out for may be cost, since this solution may well prove expensive, and if you have adopted a gradualist approach then you may not want to incur the costs of installing a complete solution all at once, and the openness of tools and RDBMS: Oracle tools, for example will not work with other databases, whereas Sybase and Ingres will.

3) Pick'n'Mix SQL
I have left this solution till last, since it is only fairly recently that it has become really feasible. It is the result of two developments in database technology:
- on the one hand, the improvement of the SQL standard and the convergence of the features available in the different RDBMS are combining, as we mentioned above, to transform the RDBMS into a "commodity" product which can be bought from any supplier;
- on the other, we have seen the emergence of development tools capable of working with multiple databases. Tools such as Uniface, PacLAN, and Powerbuilder have been well received in the user community.

To my mind these tools combine the advantages of the two previous solutions. On the one hand, they allow you to use the RDBMS which best suits your needs. Increasingly, these may prove to be the RDBMS most closely integrated with the operating system: ALLBASE on the HP, RDB on Dec, DB2 on IBM, etc. On the other, they provide a unified look and feel across multiple systems. In some cases, they may be able to reverse-engineer your existing applications which may make the migration to SQL easier.

Particularly relevant to the current HP3000 user community will be the ability of such tools to interface with IMAGE/SQL, which is our next subject.

Will IMAGE/SQL be the answer to our prayers?

Now you will have noticed that so far I have been speaking about why and how you should implement SQL databases, but have said precious little about handling the problem that the vast majority of HP3000 users are faced with: how do I start developing in SQL, when all my mission-critical applications are written using IMAGE? The number of brand new applications which do not use, or need to update, already existing data is very small nowadays, after all. This is why my previous paper a year ago came to the conclusion that you should avoid rewriting existing applications for the sake of it, and concentrate rather on pilot applications using ALLBASE. Unless you only needed read access to the old data, in which case Allbase-TurboConnect might have provided a viable solution for the new system, the only means of passing data from the old to the new systems was via transfer jobs, extracting from ALLBASE and adding to IMAGE or vice versa.

The announcement of IMAGE/SQL has radically changed the situation, and promises to have major implications. What exactly is involved?

IMAGE/SQL will allow read and write access to IMAGE databases using SQL statements (essentially Select, Insert and Update).

IMAGE/SQL will be supplied as part of FOS, to users who are paying support fees for IMAGE.

IMAGE/SQL must be implemented using ALLBASE/SQL, and the pre-processors for IMAGE/SQL are the same as those for ALLBASE.

An IMAGE database can be declared as part of an ALLBASE/SQL DBEnvironment.

What does all this mean?

First, it automatically makes ALLBASE the cheapest implementation of SQL on the HP3000 platform (ie free to all intents and purposes), which helps to make the DIY and Pick'n'Mix development options much more attractive.

Secondly, because the pre-processor is the same, any development tool or package which works with ALLBASE (eg Uniface, Powerhouse, Powerbuilder, etc) will also work with IMAGE/SQL.

ALLBASE becomes all the more attractive, in that the main interest of using IMAGE/SQL is not simply to program against IMAGE databases using SQL, but to build new applications using an RDBMS database, which can also access existing information still held in IMAGE. IMAGE/SQL allows you to do this, because it allows an ALLBASE/SQL DBEnvironment to consist of both ALLBASE tables created in the normal SQL way, and tables which are in reality IMAGE datasets. The same Select statement will be able to access data from both a relational table and an IMAGE dataset. It will even be possible to apply triggers, stored procedures, and check constraints against IMAGE datasets just as they can be applied to ALLBASE tables.

According to one school of thought, this automatically makes IMAGE into a true relational DBMS. This would mean that we can develop new SQL-based applications, and new relational databases, using IMAGE rather than one of the existing RDBMS. I do not believe that this is the case, for reasons which, again, are both economic and technical.

On the technical side, we should remember that IMAGE/SQL does not support the DDL (data definition) elements of SQL. In order to do so, think what HP would have to incorporate into IMAGE:
- ability to add new items to datasets,
- ability to dynamically increase dataset sizes,
- support for B-Tree indexes,
- ability to create and drop automatic master sets with paths to details,
- support for wild-card indexed searches,
- etc, etc.

None of this is impossible, but the end result would be to create a database file structure very similar to.... ALLBASE/SQL which exists already! Moreover, it is hard to see why HP would want to introduce into IMAGE the support for BLOBs (for example) which will be required to handle ObjectSQL in the future, when the groundwork for doing this already exists in ALLBASE.

And of course, all this has economic repercussions. Since IMAGE and ALLBASE are going to remain separate, support and development for these technologies will mean separate development teams, with all the associated costs. And of course the same would be true for users developing in IMAGE/SQL. They would have to take on board SQL knowledge, without being able to reduce their IMAGE expertise, and all this costs money.

What I have just been saying may sound pretty negative. But this is only true if we make the mistake of treating IMAGE/SQL as just another "me-too" RDBMS. In reality, we are looking at something far more exciting: a Compatibility Mode database!

Conclusion : a migration path to SQL

One of HP's great triumphs in recent years has been the move from MPE/V to Risc-based MPE/XL machines, which has been made possible by the ability to run programs written under MPE/V in Compatibility Mode. I think everyone who has been involved in migration projects will have been extremely impressed by the ease of migration and the completeness and performance of CM. This provision of upward compatibility is a real distinguishing feature for HP: to be convinced, we need only look at the "upgrade paths" between IBM Series 1, 34, 36, 38, or at DEC's present difficulties in migrating from VMS to Alpha-based systems. What HP has done with IMAGE/SQL is to offer a CM migration path from proprietary IMAGE-based systems, to SQL relational systems. The future of IMAGE is assured, not because it has been guaranteed eternal life, but because you no longer have to abandon IMAGE in order to take advantage of new technology. You can migrate there in your own time, under your own control, and step by step.

Columbus has sighted the Americas. LAND HO!

PAPER NUMBER: 7000

TITLE: Mainframe Migration Alternatives

PRESENTER: David Rubinstein
Innovative Information Systems
63 Nahatan Street
Norwood, MA 02062
617-769-7511

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Interex 9/1/93: Paper Number 7001
Distributed Transaction Processing: CICS, Encina and DCE on HP Computers

George Stachnik
CSY Planning Organization
Hewlett Packard Co.
June 30 1993

What's CICS? And What's an OLTP Monitor?

In late 1992, the headlines in the computer industry press trumpeted that something new called "CICS" was coming to HP, or more specifically to HP computers. CICS is a familiar term in IBM mainframe datacenters; it's the very backbone of the online application environment. But what will CICS bring to users of HP systems and servers? Should you be planning to install it on your HP system? The answers to these questions are complex. They involve a good deal of terminology that may be new to users of HP systems: OSF, DCE, Encina, distributed processing, the ACID properties, atomic transactions, distributed OLTP and lots more.

This paper will begin by taking a high level look at CICS, examining the world of centralized OLTP. We'll see how CICS fits into that paradigm. Next we'll look at how CICS fits into the world of distributed processing, and examine the differences between distributed OLTP and distributed computing. Then we'll take a closer look at the two software technologies that the HP versions of CICS will be based upon: OSF's Distributed Computing Environment (DCE) and Transarc's Encina products. Finally we'll look at IBM's family of CICS products and see how they will relate to HP's version. It is beyond to scope of this paper to comprehensively examine DCE, Encina or CICS. Each of these technologies is complex enough to fill books; books have already been written about each of them. We hope only to show how they relate to one another, and to introduce the user to some of the key concepts and terminology relating to each of them.

The announcement of HP's agreement to port CICS has attracted a great deal of attention among HP customers, especially those who are looking for alternatives to expensive IBM mainframes. For over twenty years, CICS has been the cornerstone of the online environment for IBM mainframe systems. The CICS acronym, which stands for "Customer Information and Control System", doesn't tell you much about what CICS does. The documentation defines CICS as an online transaction processing monitor. But exactly what does that mean?

Online transaction processing, or OLTP for short, is nothing more than a fancy way of describing what most applications do on both the HP3000 and the HP9000. OLTP is moving data back and forth between a desktop device such as a terminal, (or a PC or workstation that's acting as a terminal) and a host computer, while making certain guarantees about the integrity of that data.

An OLTP system is a computer system that supports the functionality required to do OLTP. More specifically, OLTP systems provide a way of applying transactions that meet the so-called "ACID" test. Commercial OLTP requires that the transactions have four characteristics: atomicity, consistency, isolation and durability.

* Atomicity: A transaction must be accomplished as a whole or not at all. For example, if an application program applies a transaction that causes more than one data record to be updated, then all the records must be correctly updated. If, for any reason one or more of the records cannot be updated, then the system will update none of them. In practical terms, this means that applications must have a simple way of marking the beginning and end of a logical transaction, so that any updates done in the interim will be handled in an "all-or-nothing" way.

* Consistency: The results of a transaction must be reproducible and predictable. In practical terms, this means that, given a set of initial conditions, the results of applying a transaction should always be the same, taking the data from one consistent state to another.

* Isolation: A transaction must not interfere with or be dependent upon any other concurrently executing transaction. This characteristic has also been called "serializability" and in practical terms it means that the system must do some kind of locking to ensure that only one user at a time can operate on the data affected by a particular transaction.

* Durability: The results of the transaction must be permanent. In practical terms, this usually means that the results of a transaction are committed to a storage medium (usually a disk), which will not be affected by errors, power interruptions or other system failures.

For roughly 20 years, CICS has been making it possible to do OLTP on IBM mainframes, in much the same way that MPE and its subsystems make it possible to do OLTP on an HP3000, or HP-UX and various database management systems make it possible on an HP9000. So when an IBM guru tells you that CICS is an "OLTP monitor", he's really saying that CICS makes it possible for application programs to make some assumptions about the "ACID" characteristics of their transactions.

In most cases, HP systems do not require a separate OLTP monitor like CICS because the user interfaces and other functionality required to do OLTP are built into the operating system and database management systems. That is, when a typical HP application applies a transaction to a system, it is the DBMS and (particularly on HP3000s) the operating system itself that is usually responsible for guaranteeing the ACID qualities of the transaction.

On mainframes, CICS is necessary for OLTP because IBM's mainframe architecture was designed in the early 1960s, when commercial computing meant only one thing: processing batch jobs. Just as MPE and HP-UX were designed for online processing, IBM's earliest mainframe operating systems were designed to do batch processing, and to this day that is what they do most effectively. When IBM customers began to demand the functionality required by online data processing, IBM created the entirely new and separate CICS environment, and layered it on top of the mainframe operating system.

You might wonder, therefore, why HP would want to bring CICS to HP systems, if the functionality required to do OLTP already exists in MPE/iX, HP-UX and their related DBMSs. There are two key reasons: IBM compatibility and the ability to do distributed OLTP in a heterogeneous network.

* IBM COMPATIBILITY is the first key reason why HP is excited about bringing CICS to its platforms. At this writing, there are tens of thousands of programmers that have been trained to write applications for the CICS environment. Furthermore, there is a huge portfolio of OLTP software that was designed for the CICS environment. Up until now, that software could not be used on HP systems without the use of a CICS emulator from a 3RD party.

* DISTRIBUTED OLTP IN HETEROGENEOUS NETWORKS is the second key reason why CICS is being ported to HP. CICS will make it easier for customers to distribute applications across a network that is made up of a mixture of HP systems and IBM systems. It's worth noting that CICS is not the only way (nor necessarily the best way) of distributing applications in an open environment. There are a number of other open OLTP monitors such as Top End, Tuxedo and Transarc's Encina which support the ACID qualities. But CICS is the only one that also provides the IBM compatibility re- quired by software being ported from IBM mainframes.

Distributed OLTP vs. Centralized OLTP

In the past, most OLTP applications were designed to run on a single centralized computing system. Today's dynamic business environment requires that it be possible to write OLTP applications that can be distributed over a network. Distributed OLTP means that different parts of the application run on different systems in the network and that these parts cooperate with one another to guarantee the ACID characteristics.

In the 1970s, IBM's CICS product pioneered distributed OLTP in the commercial marketplace, but with some important restrictions. Originally, all the systems in the network had to IBM mainframes; that is to say the network had to be a homogeneous network (i.e., all the systems came from one vendor and shared a common architecture). Furthermore, these mainframes had to be connected to one another using IBM's proprietary Systems Network Architecture. Finally, The applications themselves had to be designed around IBM's CICS environment.

In the late 1980s and early 1990s, open systems exploded into the commercial marketplace. One of the key factors driving the growth of open systems in the commercial marketplace has been interoperability: the ability to build heterogeneous networks of systems from many different vendors. Another factor has been application portability, making it possible to move programs from system to system relatively easily. These two characteristics made open systems the ideal platform for the development of distributed OLTP applications.

Interest in open client server solutions for OLTP is snowballing in the mainframe MIS community. Part of the reason for this is that four of the key impediments to acceptance for distributed OLTP are disintegrating before our eyes:

1) First of all, many MIS managers have perceived (rightly or wrongly) that "open systems" is just another way of saying Unix. The idea that a system can be open (by supporting open interfaces such as POSIX, TCP/IP, SQL, etc etc) without being based on Unix is only now being understood in the IBM world. IBM claims that the AS/400 is an "open" system, pointing to its support of TCP/IP, SQL and other standards. HP makes similar claims for the HP3000, adding POSIX.1 and POSIX.2 to the list of standards supported by MPE/iX. This gives MIS managers a much greater range of platforms to choose from, and does not restrict them to Unix systems alone.

2) Secondly, many MIS managers have perceived (again, rightly or wrongly) that Unix is not yet ready for the datacenter. They have questions about Unix security, availability and performance in an OLTP environment. For many, these perceptions have begun to change. Products such as CA's Unicenter have given Unix better more credibility in the datacenter while open systems that are not based on the Unix operating system (such as the HP3000 and AS/400) have gained acceptance as alternatives to MVS. The belief that MVS is the *only* operating system capable of delivering enterprise-wide MIS solutions, once widely held among MIS managers, is now viewed as a quaint idea that, if it was ever true, is certainly not true today.

3) Thirdly, open systems vendors have not been able to agree (up until now) on an industry standard OLTP monitor. That is, there has not been a single agreed upon means of guaranteeing the ACID properties of transactions that are spread across multiple systems in a network. In 1992, that changed with the adoption of Transarc's Encina as the strategic open OLTP monitor for HP, DEC, IBM and a number of other vendors. When IBM ported CICS to the Unix based RS/6000, it used the Encina toolkit as the means of guaranteeing the ACID qualities. Similarly, the Encina toolkit will be the basis of the HP implementation of CICS.

4) Finally, companies that have large investments in software for proprietary OLTP systems such as CICS have had to face up to the prospect of throwing all that software away and starting over again with open systems. The availability of CICS on open systems removes this roadblock.

Distributed Processing and Distributed OLTP

Distributed OLTP is not the same as distributed processing:

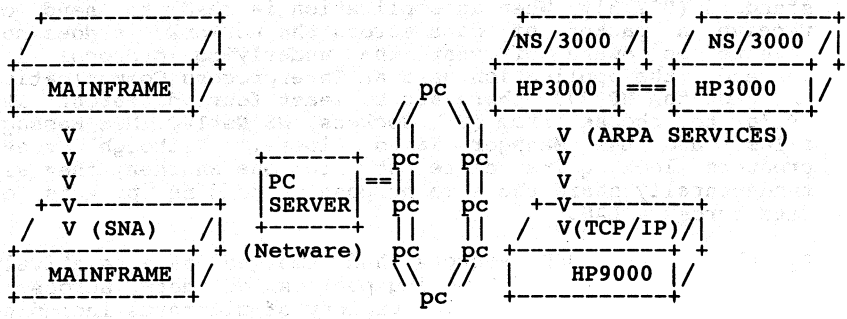
* Distributed Processing is simply the ability to tie multiple systems together in a network, and the ability for a program running on one system to access data that physically resides on other systems in the network.

* Distributed OLTP is the ability to do all of the above AND maintain the ACID properties of transactions applied to files and/or databases anywhere on the network.

The 1980s saw distributed processing graduate from being a relatively new technology, suitable for use by technologically adventurous early adapters, to becoming a mainstream technology, suitable for use by almost anyone. At the beginning of that decade, HP, DEC, IBM and many others made it possible (if not terribly easy) to distribute data throughout a homogeneous network - that is a network in which all the systems came from the same vendor and shared the same architecture.

The shortcomings of this approach became apparent in the leveraged buyout crazed business environment of the 1980s. Suddenly, many network managers found themselves faced with networks that were not so homogeneous. A company whose IT strategy had been based exclusively on one vendor's equipment might suddenly acquire a competitor that had been using another vendor exclusively. Overnight, MIS managers found themselves faced with the need to build and manage heterogeneous networks made up of systems that came from different vendors and which used different architectures.

By the early 1990s, it had become apparent that if it was going to be possible to do distributed processing in a heterogeneous (multivendor) network, then the industry would need a standard way of doing distributed processing. For example, a typical large corporation might have an SNA network for their mainframes, LAN's based on Novell's Netware for PC's, and other networks based on various protocols for midrange computers.



In the above diagram, PC's on the Netware network could access files on the Netware LAN, but could not access data on either the SNA network of mainframes, or the HP network of HP9000s and HP3000s.

Partial solutions to this problem were developed in the 1980s. Using a variety of translating, bridging and routing solutions, multiple incompatible networks could be connected to one another. The "bridge-and-route" approach to heterogeneous networks solves the problem of making data available across the network. But as is often the case, it exposed a whole new class of problems. For example, programs running on a PC might use Netware Application Program Interfaces (API)s to access data on the PC network, while programs running on the HP platforms might use a different set of APIs such as NS NetIPC. It was difficult to write applications that could easily be ported from platform to platform without having to change the APIs for network access.

Today, there are a relatively small number of industry standard network protocols that are widely in use, including the Transport Control Protocol (TCP), which originated with Unix systems and which is widely used by non-Unix-based computers as well, and Ethernet, which is based on an IEEE standard (802.3). When an application is ready to send or receive a packet of data across the network, it does not have to be aware of what the underlying protocol is. Instead, the application uses an Interprocess Communication API. On the HP3000, there are at least four different IPC API's to choose from (BSD Sockets, NS NetIPC, MPE Message files, and LAN Manager Named Pipes). Although these products look quite different from one another, they all fundamentally share the same purpose - sending packets of data across a LAN.

By the 1990s, BSD Sockets had emerged as a relatively standard set of APIs for moving packets of data across a network. Supported on a wide variety of platforms including the HP3000, DEC VAX, IBM AS/400, DOS PCs, and for most networks including PC/LANS, Ethernet, TCP/IP and many others, BSD Sockets represents an open way to do distributed processing. It is the foundation of many client/server applications today, in which the client, typically a desktop PC or workstation, sends a packet of information to a data base server, where another program accepts the packet and uses it to read or update a data base before sending a packet containing the results back to the client.

But once again, the solution of one problem served to expose yet another one. MIS managers soon found that they wanted to be able to use any kind of client with any kind of server. Unfortunately, different platforms sometimes represent data internally in radically different ways. For example, data is typically stored in ASCII on HP computers, but in EBCDIC on IBM mainframes. Different hardware architectures store binary data in different ways, sometimes referred to as 'big-endian' and 'little-endian'. What happens when an HP client sends a packet of ASCII data to an IBM mainframe that uses EBCDIC?

BSD Sockets and the "translate-bridge-and-route" approach to heterogeneous networks do not address these issues - they simply move packets of bits across the network and assume that the applications will correctly interpret the bits. So something more than an IPC package is required to do distributed processing in a truly open fashion. And that something more is called "DCE".

Distributed Processing in Heterogeneous Networks: DCE

One of the organizations that undertook the creation of a standard for open distributed computing was the Open Systems Foundation, or OSF. OSF is made up a large number of vendors of computer equipment, networks, software and related products. OSF counts as its members HP, IBM, DEC, Microsoft and many others, as well as many of the world's largest users of these products. By 1992, OSF had agreed on a set of standards for distributed computing collectively called the "Distributed Computing Environment", or DCE.

There are five parts to the DCE standard:

1) Remote Procedure Calling, or RPC.
2) Security
3) Timing Services
4) Naming services
5) Directory services

It is beyond the scope of this paper to cover DCE in detail. But we will explain some of the key DCE features that are used by CICS.

The basic unit of operation in a DCE network is called a "cell". A cell is defined in OSF's document "Introduction to OSF DCE" as "a group of users, systems and resources that are typically centered around a common purpose and that share common DCE services. At a minimum, a cell configuration includes one Cell Directory Server, one Security Server and one Distributed Time Server. A cell can consist of from one system to as many as several thousand systems. Systems in the cell can be in the same geographic area (for example, on the same LAN), but geography does not necessarily determine a cell's boundaries. The boundaries of a cell are typically influenced by its purpose, as well as by security, administrative and performance considerations.

Remote Procedure Calling

The backbone of DCE is its ability to do remote procedure calling, or RPC. RPC makes it possible for applications to do procedure calls across the network. That is, an application program running on one computer can call a procedure that is executed by a different computer. If data is stored in different formats on the two computers, RPC will ensure that the necessary translation is done, so that the differences are transparent to the program that calls the procedure.

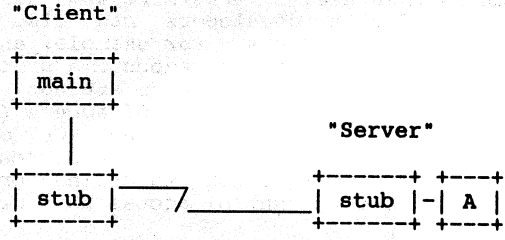
To understand how RPC works, consider an application made up of a main procedure and a second procedure named A. "Main" uses ordinary procedure calls to cause A to be executed. In an ordinary, host-based environment, both "main" and "A" would reside on the same computer system. We could therefore be certain that both "main" and "A" would represent data in exactly the same format - ASCII, EBCDIC, big-endian, little-endian or whatever.

Suppose we wanted to split application into a client server application, with "main" executing on a client system, and "A" executing on a server. We could use BSD sockets to pass data back and forth between "main" and "A". But if we took that route, we would have to write code to handle any differences in the how the client and server represent data. For example, a program running on an client would need to know if the server was using ASCII or EBCDIC, and be prepared to do a translation if necessary.

DCE handles this situation through the use of an "Interface Definition Language" or IDL. IDL programs are generally quite short. They are made up of definitions in a C-like syntax of the names of each of the procedures in the application, the names and formats of the parameters that are passed to the procedure, and other DCE-specific details, such as what to do if the network fails to respond to a procedure call within a specified time.

In our example, an IDL compiler would be used to process an IDL file containing descriptions of "main" and "A", along with the source code for both procedures. The IDL compiler would then generate short "stub" programs for use on both the client and the server. These are then used together with a header file to build the client server application.

In the DCE version of this application, when "main" calls "A", the IDL compiler redirects the call to the stub, which resides on the same machine as "main". The stub then sends a packet of information to the server, which invokes the procedure "A", using its stub on the server. When "A" has finished executing, it returns control (again via the stubs) across the network to main.



A key advantage to doing things this way is that the IDL and the stubs that are generated handle architectural differences that may exist between the client and server. For example, if the client uses ASCII, and the server uses EBCDIC, the stubs could handle the translation. The important thing to understand is that when "main" calls "A", it doesn't necessarily have to be aware what kind of server "A" is running on.

DCE Security using ACLs and Kerberos

DCE also defines a set of standards for managing the security of a network. Access Control Lists (ACL)s and Kerberos Authentication Services make it possible for the network administrator to exercise controls over what files, data and resources are available to programs, regardless of which nodes in the network they are running on.

Access Control Lists should already be familiar to HP computer users. They have been available on both HP-UX and MPE/iX systems for some time now. (MPE/iX refers to them as as Access Control Definitions, or ACDs, rather than as ACLs. But the concept is the same). Briefly, an ACL is a list of who has access to a particular file or directory, and what kinds of access are permitted.

In a DCE environment, the concept of ACLs (or ACDs, as you prefer) has been extended further to apply to any DCE-defined object in the network. For example, a printer that is attached to a print server may be associated with an ACL. The ACL is then used to determine who has access to the printer. Software developers can also use ACLs to control access to DCE services. For example, an ACL might control who has permission to execute a procedure (using RPC) on a server. Software developers are not restricted to being able to use only a few kinds of access (Read, Write, Execute, and so forth) in ACLs. The ACL mechanism is flexible enough to allow the definition of "custom" access types: for example one might want to define "print access" if ACLs are being used to control access to a printer.

ACLs are the DCE tool for controlling who has access to various network resources. DCE also supports a means of authenticating users who simply wish to logon to the network, and to systems on the network: this authentication service is called Kerberos. Kerberos authentication services are used to validate passwords. This is a particularly thorny problem in a networked environment, because transmitting passwords in "clear text" (i.e., not encrypted) across a network is an invitation to the enterprising hacker. Ordinarily, data that is transmitted across a network is available to every node on the network. Kerberos ensures that sensitive data is protected from prying eyes using a data encryption scheme.

Once again, it is beyond the scope of this paper to provide a detailed description of Kerberos - but some of the key elements are:

* One of the nodes that is attached to the network is identified as the security server. This server maintains a copy of every user's password in a database called the "Registry". Network access is not permitted unless this password is matched. The security server and registry must be physically secured against unauthorized access.

* Passwords are not transmitted in clear text between nodes on the network. When a user wants to logon to the network, the request is passed to the security server, which begins an encrypted dialog of messages with the client, involving "Ticket Granting Tickets" and "Privilege Attribute Certificates". The user is granted access to the network, and to resources known to DCE based on ACLs and authorization by the Kerberos security server.

Without Kerberos, security is almost always a concern in a networked environment, because sensitive information (such as passwords) that is transmitted across the network is available to every node on the network. It is simply a matter of programming to enable a node to eavesdrop on this sensitive information. With Kerberos, this sensitive information is transmitted in an encrypted fashion, using DES or other encryption techniques.

Other DCE Services

DCE timing services make it possible for all the nodes in the network to synchronize their clocks - so that when applications are distributed across the network, they do not have to worry about the time being different on different nodes. This is an important service for OLTP applications that "time stamp" the data in their transactions. Each DCE cell includes a Distributed Time Server for this purpose. Similarly, DCE provides directory services, managed by a Cell Directory Server, for keeping track of the cells in the DCE network.

DCE standards provide a framework that will makes it possible to write distributed applications that run on a network of systems. The systems may be incompatible with one another at a hardware level, at an O/S level, and even at a data storage level. But as long as they all provide support for DCE, then an application can be distributed over multiple systems, and the pieces of the application can be made to work with one another in a predictable way.

Distributed OLTP: Encina

DCE standards are not sufficient to support distributed OLTP. Nowhere do the DCE standards address the ACID properties that are required for distributed OLTP transactions. DCE can be used to apply all the parts of a transaction to a network of systems. But should one of the parts of the transaction fail, the application itself would be responsible for identifying the failure and backing out the parts of the transaction that were applied successfully. This greatly increases the complexity of the application. It makes more sense to build this functionality into a separate layer of software - an OLTP monitor.

There are a number of OLTP monitors that are currently available for Unix-based systems (such as Tuxedo or Top End). However, CICS in particular is built on top of a product called "Encina" which comes from a company called "Transarc".

Once again, it is beyond the scope of this paper to provide a comprehensive look at Encina, but we will look at the parts of Encina that relate to CICS. Encina is based on DCE, and takes advantage of DCE services such as RPC, ACLs and Kerberos. Encina clearly divides the application into client software (which typically has only display functions) and server software. Encina defines an application server as a system which executes server software. The client connects to the application server and submits the transaction data. The application server executes the steps required to accomplish the transaction using a DBMS-like resource manager.

Encina adds two tiers of software on top of DCE: the Base TP Services, and the extended TP Services. The Encina Base TP Services are made up of two components: the Encina Toolkit Executive, and the Encina Toolkit Server Core. These are the parts of Encina that make it possible to make transactions span a network without losing their atomicity.

The Encina Toolkit Executive includes a set of application program interfaces (API's) called "Transactional-C". Software developers can call these API's to mark the beginning and end of logical distributed transactions. The following is a simple example of transaction definition using transactional-c.

```
transaction {...
     debit(salaryExpense,amount);
     credit(accountsPayable,amount);
     enterAuditData(employeeIdentifier,amount,date);
     ...}
onCommit
     printf("Transaction succeeded.");
onAbort
     printf("Transaction failed.");
```

As you can see, the transaction is made up of three parts: a debit, a credit, and the entry of some auditing data. These three parts are bracketed into a transaction which is committed guaranteeing the ACID qualities. Note that the application program does not need to have any special code to handle transactions that abort - it is all handled automatically by Encina. Note also that the program must use the "transaction {...}" syntax to define the beginning and end of the transaction.

Just as DCE used an IDL to define the various parts of a distributed application to one another, Encina uses an extension to this approach called a Transactional IDL (TIDL) to adapt DCE's RPC mechanism to a transactional environment.

The Encina Toolkit Server Core provides services such as locking, and an interface that's compatible with X/Open's XA interface for use by Data Base Management Systems (DBMSs) and other products...

A second tier of Encina products and services include:

* The Encina Structured File Server (SFS)
* The Encina Peer-to-Peer Communication Services (PPC)

The Structured File Server

Encina's Structured File Server (or SFS) manages data stored in record-oriented files on a Unix-like file system. SFS differs from the standard Unix file system. The Unix file system recognizes only one kind of file: a so-called "byte stream file". These files are simply a stream of bytes - there is no record structure such as is found on commercial operating systems such as MPE or MVS or on relational databases.

SFS stores information on volumes - which are logical units of disk storage that can be managed individually. Volumes can consist of single or multiple partitions, and can include an entire disk, or can span multiple physical disk devices.

SFS files are formatted into records of a pre-defined maximum size and format. Records are subdivided into fields. SFS files are of three different types:

* Entry Sequenced (access is sequential)

* Relative (access is by relative record number)

* Clustered (access may be sequential or random - by key)

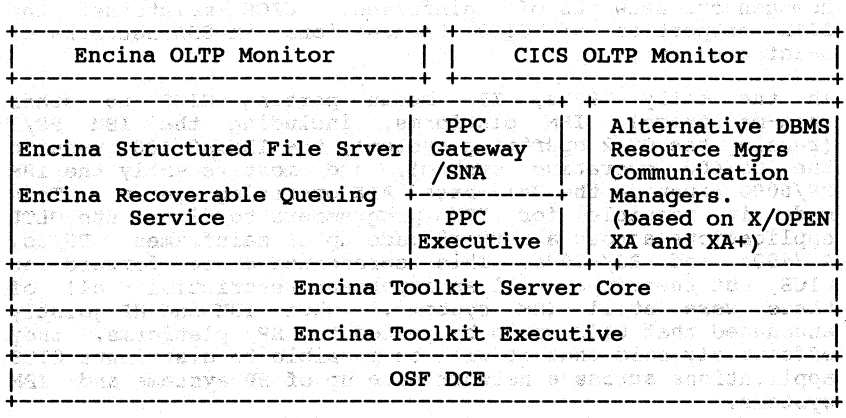
When SFS files are opened, Encina's SFS assigns an Open File Descriptor (OFD). This may be "transactional" or "non-transactional". If a non-transactional OFD is selected, then there is no guarantee that the results of operations against the file will be permanent in the event of a media or system failure. But if a transactional OFD is selected, then changes to the file must be made in the context of transactions (defined using the transactional C language mentioned earlier). SFS guarantees that these transaction changes are made permanent when each transaction commits, and SFS will roll the changes back if the transaction aborts for any reason. SFS uses a log-based recovery scheme that permits fast restarts in the event of errors. It was designed for 2-phase commit protocols, which allows multiple SFS servers to be used within a single transaction.

Encina peer-to-peer Communication (PPC) Services

This service supports transactional peer-to-peer communication using either TCP/IP or SNA (LU6.2) transports. Encina PPC services deliver LU6.2 sync-level 2 services (i.e. 2-phase commit). The PPC Executive makes it possible for Encina to carry on transactional peer-to-peer conversations over TCP/IP. The PPC Gateway works with the PPC Executive to communicate over SNA/LU6.2 to mainframe software that supports LU6.2, such as IBM's CICS/ESA.

What's the Relationship Between CICS and the Encina Products?

The following diagram shows how CICS and the components of Encina fit together. The Encina OLTP monitor is built on top of the Encina toolkit (which includes SFS, RQS, and the PPC products). It allows applications running under the control of the monitor to apply transactions while maintaining the ACID qualities. Using industry standard interfaces such as X/Open's XA and XA+, other database managers such as HP's ALLBASE can also participate in these transactions. The Encina Monitor maintains a database of all application servers and coordinates Encina operations with other Encina monitors running elsewhere in a DCE network, in order to do load balancing.



CICS represents an alternative to Encina's OLTP Monitor. It uses many of the the technologies that are part of Encina and DCE, such as SFS.

Although CICS can be thought of as an alternative to the Encina OLTP monitor - one that provides a level of compatibility for applications that were originally written for the mainframe-based CICS system, it would be a mistake to think of CICS as being "just an OLTP monitor." CICS also has components that relate to the user interface and to resource management. CICS application programs are executed under the control of CICS, and cannot be used in other environments without extensive modifications.

CICS Products

In the past few years, CICS has gone from being a wholly proprietary "mainframe only" product to being a relatively open product that's available on a number of different platforms. In the 1970s and 1980s, a CICS transaction running on a mainframe could update multiple files or databases - or multiple kinds of files and databases - or files and databases spread across multiple mainframes in a network - while maintaining the ACID properties of the transaction - but with one important restriction: all the files and databases had to reside on mainframes running CICS. The mainframe version of CICS was only useful in homogeneous networks of mainframes. CICS maintained the ACID properties of CICS transactions in SNA networks of mainframes only.

In the early 1990s, IBM began porting CICS to other (non-mainframe) IBM platforms, including the IBM PS/2 (running the OS/2 operating system), the IBM AS/400 (running the OS/400 operating system), and most recently the IBM RS/6000 (running the Unix-based AIX operating system). This made it possible for IBM programmers to distribute OLTP applications across a network made up of mainframes, PS/2s, AS/400s and RS/6000s. This represented a step forward for CICS, but there was still an important restriction - all of these were still IBM systems. When IBM and HP jointly announced that CICS would be ported to HP platforms, they effectively said that it will be possible to distribute CICS applications across a network made up of HP systems and IBM systems.

At this writing, there are no fewer than eight CICS products available or in development:

Product Name	Vendor Release	Avail. Date	Comments
CICS/ESA	IBM 3.3	3/92	Runs on ESA operating system and ES/9000 hardware. Supercedes earlier CICS/MVS product, which is used on older IBM 43xx, 30xx, S/370 systems.
CICS/VSE	IBM 2.2	3/93	Runs on VSE operating system, which is primarily found on smaller 43xx and 30xx mainframe hardware.
CICS OS/2	IBM 1.20	Avail	For use with PS/2 desktop workstations and IBM PCs using PC DOS.
CICS/400	IBM 2.2	3/93	For use with IBM AS/400 midrange systems.
CICS/6000	IBM 1.0	6/93	For use with IBM RS/6000 systems using the Unix based AIX operating system. Does *not* run under AIX ESA or AIX/PS2. It is compatible with CICS/MVS release 2.1 with some restrictions.
CICS/VM	IBM Rel 2	Avail	This older product was withdrawn at the end of 1992. It ran on the VM operating system, used on some older mainframe systems.
CICS/9000	HP 1.0	12/93	Based on CICS/6000 release 1.0
CICS/3000	HP 1.0	'94	

HP's CICS products will be based upon the IBM CICS/6000 product, which in turn is compatible with CICS/MVS release 2.1 (with some restrictions). For example, CICS/MVS has traditionally supported two separate sets of APIs: the command level, intended for use in high-level languages such as COBOL, and the macro level, intended for use in S/370 assembler language programs. CICS/6000 provides support for the command level only. (Similarly, support for macro level APIs was dropped from CICS/ESA as of release 3.3...)

Like CICS/MVS, CICS/6000 maintains the ACID properties of distributed transactions. Unlike CICS/MVS, it does not maintain those properties using proprietary technologies built into CICS. Instead, CICS/6000, CICS/iX and CICS/UX are all being built on top of Transarc's Encina and OSF's DCE technologies. Appendix A includes a more detailed explanation of some of the differences between CICS/MVS and the CICS Reference port that forms the basis of both CICS/6000 and the HP versions of CICS.

Application Design and Transaction Demarcation in CICS

CICS application design begins by understanding the business requirements and designing the external interfaces. There are two ways in which a CICS/6000 application can interface with an end user: using a dumb terminal character-based interface, or using a graphical user interface.

CICS application programs use a subsystem called "Basic Mapping Support" (BMS) to communicate with terminals and map the data onto the terminal screen in a formatted way. BMS includes facilities to define how data will be laid out on the terminal screen, and commands to move data between the program and the screen. BMS uses short programs called BMS map definitions to define the screen layout. These maps can be coded using three BMS macros:

DFHMSD - to define a map set

DFHMDI - to define a map

DFHMDF - to define a field.

Coding screen layouts in this way can be somewhat tedious - therefore CICS provides a means of creating the maps interactively using a tool called the Screen Design Aid (SDA). (Users of the HP3000 VPLUS system will find some similarities between SDA and the MPE-based FORMSPEC program. Users of client/server application development tools such as Powerbuilder from Powersoft will also find SDA to be familiar). BMS maps are incorporated into the application program through the use of a tool called "cicsmap", which generates header files that can be used with COBOL or C application programs.

BMS insulates CICS applications from the end user devices such as terminals or workstations. The application programmer does not need to worry about what kinds of devices might be used to invoke the application or communicate with it.

In the CICS environment, the programmer writes an application in the form of small simple programs called transactions. These transactions can only run under the control of CICS in a CICS region. They cannot be run independently of CICS. On mainframes, it is not uncommon for multiple copies of CICS to be executing simultaneously in multiple regions. In this way, CICS regions can be dedicated to particular applications.

A CICS transaction contains the logic required to apply one logical transaction to the system, and then terminate. There is no need for the programmer to code "brackets" around the transaction, because the program *is* the transaction. None of the updates done by the program are committed until the program terminates.

This is quite different from the model used by many HP3000 and HP9000 applications - which contain a loop that is executed once for each transaction entered by the user. Such applications must include logic to bracket the beginning and end of each transaction, and to commit the transaction and make it permanent. (See the transactional-c example a few pages back.) In CICS the act of terminating implies that the transaction is to be committed and made permanent.

CICS Application Design and Resource Management

There are two major "styles" of CICS application design: conversational and pseudo-conversational. In the conversational style, the CICS task sends data to the screen and then waits for a reply. In a conversational transaction, the program that is executing may lock system resources, and hold those locks throughout the "conversation" that it is holding with the end user. This can be devastating to system performance, particularly if the user's "think time" includes a trip to the lunchroom.

In the pseudo-conversational style, transactions are broken up into tasks. End user dialogs do not span tasks - and all locks on system resources are released whenever a read-operation is posted on the user's terminal. Thus if the user decides to go to lunch while an application is waiting for input, you can be certain that the application will not have tied up valuable system resources that may be holding up other users that are waiting for them.

One of CICS's strong points is the way it enables the system administrator and application programmer to manage system resources, including application resources such as programs and files. The following table identifies some of the classes of resources that are managed in a CICS environment through the use of a program called SMIT: the System Management Interface Tool

CD	Transaction Definition	CICS uses the TD information to identify and run trans- actions. The TD information includes the name associated with each transaction (transaction identifier), its status (enabled or disabled), identifiers that indicate where the transaction is physically located in the network, as well as information relating to security and prioritization.
PD	Program Definition	CICS program definitions identify the programs, BMS map sets and tables that are used in a particular CICS region. Each of these objects is associated with a program identifier, and (if the program resides in a remote region or on a remote system) the name of the region or system where it resides.
WD	Terminal Definitions	The CICS terminal definitions define the kinds and config- urations of terminals that are available in this CICS region.

CD	Communication Definitions	CICS communication definitions define the other systems with which this region can communicate. They also define the connection types (TCP/IP or SNA) used with other systems.
UD	User Definitions	This definition is used to perform transaction and resource security checking. CICS uses the DCE principal id to search for and validate users. It also contains information relating to user priorities and security keys for particular transactions.
FD	File Definitions	On mainframes, CICS is tightly integrated with VSAM. In the open implementation of CICS, the Encina SFS is used instead. SFS files map closely into the IBM VSAM filetypes such as Entry Sequenced Data Sets (ESDS) Relative Record Data Sets (RRDSs) and Key Sequenced Data sets (KSDSs). The CICS file definitions identify and describe the files that are available for use by CICS applications running in this region.
TSD	Temporary Storage Queue Definitions	A "Temporary Storage Queue" is a programmatic "scratch pad" area that can be used without any need to define them in advance. TSDs can reside on the local system, or on a remote system, and they can be made recoverable under some circumstances.
TDD	Transient Data Queue Definitions	Transient Data Queues also represent a kind of "scratch pad" for use by applications, but unlike TSDs, TDDs must be pre-defined.
JD	Journal Definitions	CICS journals are sequential files used to hold data that is used to reconstruct events or data changes. Journals may be used as audit trails or as transaction records.

RD	Region Definitions	The Region definitions include the parameters that are used when initializing CICS, such as automatic startup parameters, programs to be executed automatically at startup and shutdown, a region identifier and so forth.

CICS, VSAM and Encina

It is not my intent in this paper to do a feature by feature comparison of Encina and CICS. However, some contrasts between the two products are inescapable.

Encina has an advantage over CICS in that it represents an industry standard. It has been agreed to by multiple vendors, including HP, DEC and even IBM. It's available today on a number of platforms. But Encina also has the disadvantage of being very new. There are few applications today that have been written around its industry standard interfaces.

CICS, on the other hand, has been around since the early 1970s. Most online applications that operate on mainframes use it - and as CICS begins to make its appearance on HP platforms, those applications will surely follow. However, there are still questions about whether or not even IBM feels that a Unix-based version of CICS is ready to take on the large-scale OLTP demands associated with mainframes. As of January of '93, IBM was still recommending against deploying initial releases of CICS/6000 in full scale production environments. HP announced its implementation of DCE in 1992 for the HP9000, but as of this writing, neither Encina nor CICS are shipping on HP systems.

The road from mainframe CICS to Unix-based CICS may not be smooth. Gartner Group analyst Roy Schulte estimates that only 28% of today's CICS applications could be ported to Unix cost-effectively, and that the remaining 72% are old programs using Assembler or macro-level CICS, which are not supported in the AIX or HP versions.

In this paper, I've tried to explain some of the things that CICS has been used for in mainframe installations. I've also tried to explain a few of the basic concepts and problems associated with distributed OLTP.

Appendix A:

The following table documents some of the key differences between mainframe CICS and the CICS reference port that forms the basis of HP's CICS products:

General Differences

HP CICS Reference Port	MVS Version of CICS
BMS and Terminal Support Basic Mapping Support function supports all 3270s and SCS printers.	Added function supports non-3270 devices and BMS paging functions.
Double Byte Character Support (DBCS) for map field definitions.	No DBCS support for DFHMDF macro.
BMS BLOCK definitions not supported.	BMS BLOCK definition supported
Terminal shell can't emulate all 3270 devices - Specialized device support (e.g. X-terminal window) may be added.	Terminal function supports all 3270 devices.
No sequential terminal support (Better alternative is to use the Program List Table)	Seq'l Terminal supported, but most shops use PLTPI as a better alternative...

Application Programming Interface (API)

CICS Reference Port	MVS Version of CICS
Command level APIs are supported. Older macro level APIs (used on MVS systems in assembler) are not supported.	CICS/MVS supports Macro API CICS/ESA 3.3 supports cmd level API only
Interval Control WAIT EVENT and POST not supported	Support for WAIT EVENT & POST to synchronize event completion.
Task control SUSPEND releases control to the operating system.	Task Control SUSPEND releases control to CICS task mgt: (DFHKCP).
Task control NOSUSPEND option not operational	NOSUSPEND supported.

Restart and Recovery

CICS Reference Port	MVS Version of CICS
Automatic journalling of CICS resources not supported. Support implied with forward recovery facilities of Transarc SFS	Automatic journalling post update record images) supported.

Intersystems Communications

CICS Reference Port	MVS Version of CICS
DPL (Distributed Program Link) ISC function provided.	NO DPL support in CICS/MVS Yes in CICS/ESA.
Batch Data Interchange not supported	Batch Data Interchange supported to interface with IBM DC systems.
No DL/I services.	DL/I (IMS) services supported...

References:

"CICS/6000 EPP Training" copyright IBM Corp & Skill Dynamics 10/25/92

"CICS/6000 and AIX OLTP T3 - Application design, preparation and implementation" - 10/15/92 copyright IBM Corp - International Technical Support Center - Austin, TX and San Jose, CA.

"Distributed Computing Environment" by Joseph Berry copyright Interact Magazine - Volume 12; Issue 8 - August '92 copyright Interact Magazine - Volume 12; Issue 10 - October '92

"Encina From Transarc. Enterprise Computing in a New Age" copyright 1991 by Transarc Corporation: The Gulf Tower; 707 Grant St. Pittsburgh PA 15219

"HP announces DCE Development Environment" copyright Digital News & Review 12/7/92

"IBM gives OLTP A New Look" by Julia King. copyright Corporate Computing 1/93 Volume 2 number 1.

"OSF Distributed Computing Environment Rationale" copyright Open Software Foundation 11 Cambridge Center, Cambridge MA 02142 5/14/90

"Rapid Development of Client-Server Applications using RPC" by Charles Knouse. copyright Interact Magazine - Volume 13; Issue 2 - February 1993

"Transarc Encina - Will Distributed OLTP Systems Overrun the mainframe's last stronghold?" From the Distributed Computing Monitor 11/92 v7 #11. copyright Patricia Seybold's Office Group

PAPER # 7002

ECONOMIC JUSTIFICATION OF A SYSTEM UPGRADE

Narendra K. Sharma
Operations Manager, Information Systems Department

Meriter Hospital, Inc.
202 South Park Street
Madison, WI 53715
(608) 267-6073

ABSTRACT:

Under many circumstances in today's competitive world economy, financial justification of projects based on net present value analysis alone is insufficient. A more comprehensive project analysis, including an assessment of implicit costs and other risk factors, is required to fully understand a project.

Accounting costs include the monetary value of transactions at the time they occur. These historical costs do not present the current economic value of the goods or services in question. Alternatively, the economic definition of cost is derived for the purpose of managerial decision making. Economic costs include the present and future costs of resources that could be allocated to other activities.

Using the example of evaluating alternatives for a processor upgrade, this paper illustrates a model using decision tree analysis in addition to financial justification to determine the preferred alternative that will best meet the organization's needs.

I. INTRODUCTION:

Many factors may contribute to the need for additional computing resources in an organization:

* Increased data file sizes and unacceptably lengthy processing times
* Rapidly increasing demands for department initiated reports
* Overhead created by system enhancements
* Increased number of batch reports
* Decreased user/customer patience for system down-time for scheduled and required maintenance
* New applications

As part of the justification for upgraded processing capacity, information systems managers are also faced with questions like:

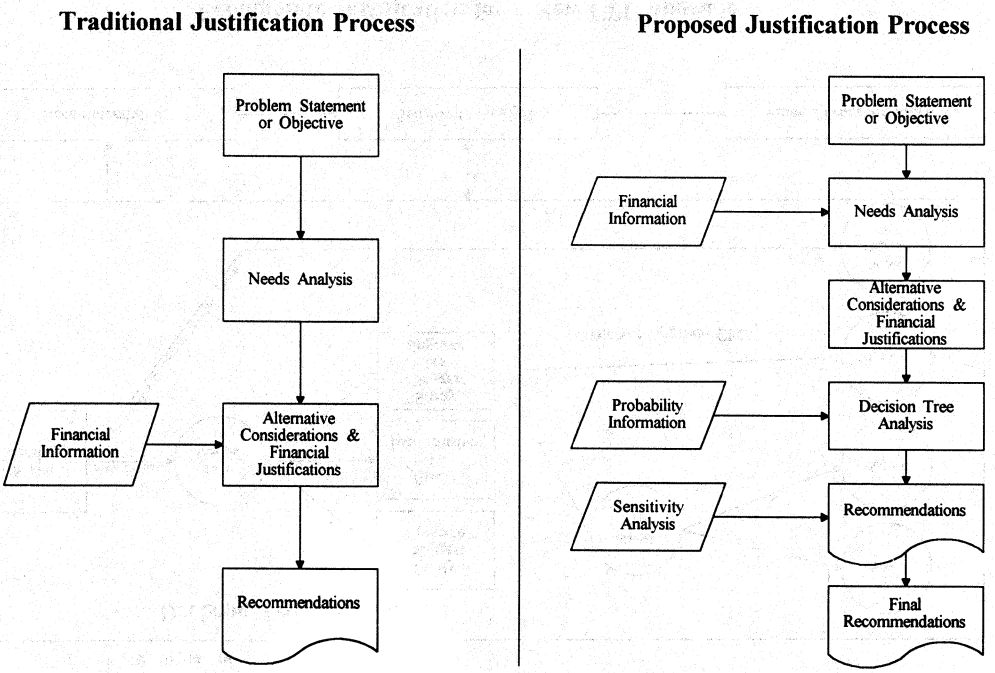
* What are the operational and financial impacts of acquiring a CPU upgrade?
* How long will a new CPU serve the organization?

The efficient allocation of the scarce resources of an organization is also a primary concern for today's IS manager in order to maximize the value of the enterprise, and the planning for addition of computing resources must take into consideration additional factors beyond the technical analysis.

This paper presents a cost/benefit analysis methodology for a CPU upgrade cost justification when a CPU bottleneck has been established, that assesses the desirability of projects when it is necessary to take both a long and a wide view of the repercussions of a particular program expenditure or policy change, and seeks to measure all economic impacts of the project, side effects as well as direct effects.

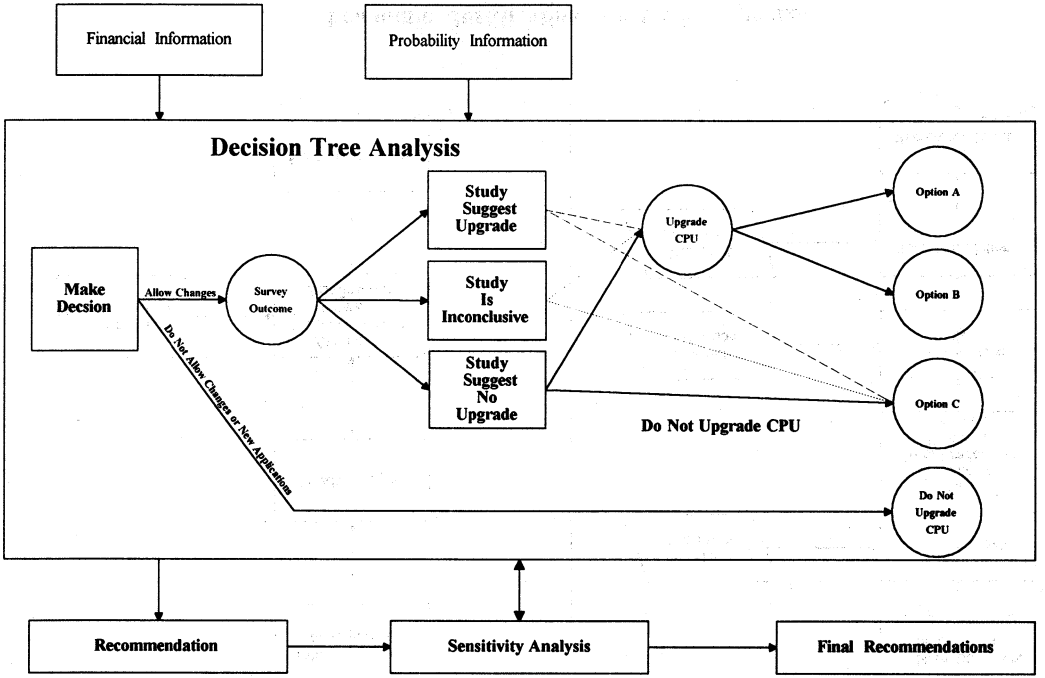
II. THEORETICAL DISCUSSION:

In order to meet customer service expectations, several alternatives are usually evaluated. Assume Option-A and Option-B represent two different CPU models that you are interested in evaluating, and Option-C represents keeping the current system. Which option would incur the least cost (or provide the best payoff) within the next five year period for the organization?



Economic Justification For A CPU Upgrade

7002-2A



Economimc Justification for a New CPU upgrade

7002 - 2B

1. STATISTICAL DATA COLLECTION AND RESULTS

Prior to the analysis, statistical data collection is required. A brief review of probability follows:

A. Probability

Most decisions are made in a condition of uncertainty where a perfect forecast of the future is unavailable. In this situation, the mathematical theory of probability furnishes a tool for the decision maker.

In any random experiment there is always uncertainty as to whether a particular event will occur or not. As a matter of chance or probability with which we expect events to occur, it is convenient to assign a number between 0 and 1. The probability indicates the likelihood of a given event occurrence.

Two fundamental statements about probabilities:

A. Probabilities of all of the various possible outcomes of a trial must sum to one.

B. Probabilities are always greater than or equal to zero and are less than or equal to one.

Marginal Probability:
Marginal probability refers to the probability of the outcome of an event not conditional on the occurrence of other event.

Conditional Probability:

$$P(B|A) = \frac{P(B \text{ and } A)}{P(A)}, \quad \text{where } P(A) > 0$$

The conditional probability of event B given that event A has occurred, is equal to the joint probability of A and B divided by the probability of event A. Or simply stated, a conditional probability is a probability of occurrence of some event, given that some event is known to have occurred.

Joint Probability:
Joint probabilities are obtained by multiplying the marginal probabilities by the conditional probabilities. Thus, joint probabilities are the probabilities of

the occurrence of two or more events.

Posterior Probability:
A prior probability of the occurrence of an event may be revised using Bayes theorem to produce a revised or posterior probability.

Prior Probability:
Prior probability is the probability guessed by an individual for a possible outcome of an event. Prior probabilities are precisely the original probabilities (or gut feeling) obtained prior to any survey information is obtained.

B. Statistical Analysis

End users are surveyed for this analysis. Given that whatever information may be available to them regarding future system enhancements, day to day workload, current system response time and their perceived need, they need to be asked to check one of the possible three possible categories for decision making for each type of system usage (high system usage or low system usage). They need to be asked,

(1) If the system usage is high (High CPU utilization, slower response time), do they believe that their
a. Computation needs will increase by more than fifty percent or,
b. Computation needs will remain same (change less than fifty percent up or down or,
c. Computation needs will decrease by more than fifty percents.

(2) If the system usage is low (Low CPU utilization, higher response time), do they believe that their
a. Computation needs will increase by more than fifty percent. or,
b. Computation needs will remain same (change less than fifty percent up or down. or,
c. Computation needs will decrease by more than fifty percent.

Based on the users conditional probabilities of software change requirement and a new system implementation for a given potential usage level, a joint probability table is constructed. The marginal probabilities of system usage (high or low) and the marginal probabilities of survey predictions (study suggesting upgrade, i.e. new computation needs will increase by more than fifty percent = X; study results are inconclusive, i.e. new computation

needs will remain same - change less than fifty percent up or down= Y; and study suggests no upgrade, i.e. new computation needs will decrease by more than fifty percent = Z) are calculated. The posterior probabilities of particular levels of system usage (high or low) for all options are also calculated.

2. FINANCIAL DATA AND ANALYSIS:

Financial information for the respective options is calculated for five years. Each option considers costs under conditions of both high and low system usage. This is done to assure that costs of the resources attained and not utilized include opportunity costs.

Following is an example of relevant costs for the financial analysis.

A. Yearly maintenance cost (net present value for 5 Years).
B. Present worth of annual HP maintenance costs.
C. Present worth of annual organization specific maintenance costs.
D. Multiple CPU vs single CPU advantages.
E. Lost strategic advantage cost, increased operational cost (if upgrade doesn't occur)
F. Price for each option (net cash out flow)

The total cost for each option is calculated. These costs will later be used in the decision tree analysis as a payoff for a given option.

3. DECISION TREE ANALYSIS:

A decision tree can be described as a graphic tool for describing the actions available to the decision maker, the events that can occur, and the relationship between these actions and events. Decision tree analysis is useful for modeling a problem as well as finding a solution. Decision tree analysis incorporates probability information as well as financial information to provide a solution or set of solutions. Based on probability, statistical and financial information, decision tree analysis is performed to draw conclusions and to make further recommendations.

4. SENSITIVITY ANALYSIS:

Sensitivity analysis demonstrates the effect of changes in one of the decision variables. The aim is to see how the performance measure is

effected.

Probability simulation is done to see the effect of conditional probability on the payoff table, corresponding changes in the posterior probabilities and decision making process. Similar analysis can be performed for payoffs to find out the sensitivity point where the payoff for two decision nodes is equal. This will help assure integrity of a solution recommendation.

III. CASE ANALYSIS - A PRACTICAL EXAMPLE:

Note: The financial and statistical information is hypothetical and intended for illustration purposes only.

1. Introduction

Meriter hospital is a 517 bed, two hospital campus facility located in Madison, Wisconsin. The hospital is currently (at the time of writing this paper) using a Hewlett Packard (HP)3000/960 (CPU-A) System to execute Census Registration, Medical Records and Patient Accounting applications. In addition, an HP3000/950 (CPU-B) System is used for Order Communication and Results Reporting applications. Both systems operate under Hewlett Packard's MPE/XL version 3.1 Operating System and application software developed by Gerber Alley(First Data) Corporation. This paper recommends an economic justification methodology using decision tree and sensitivity analysis when a CPU bottleneck has been determined.

Clinical lab applications run on an HP3000/957 using ALS application software. The pharmacy operates on Data General Aviion series system using Meditech Software and the financial applications (Human resources, Payroll, Accounts Payable, Purchasing, Inventory control and General Ledger) run on an IBM ES9000 Series System. None of the systems listed in this paragraph are expected to be upgraded in the near future.

The Gerber Alley systems are reaching a threshold where dramatic upgrade will be required. The problem Meriter is expected to face will result in delayed report distribution caused by increased batch completion times. End users will not be able to receive reports until 07:30 A.M.. Most of the report recipients are working at Meriter by 07:00 to 07:30 A.M.. The increased file size, insufficient CPU (central processing unit) power and increased system overhead will necessitate a CPU upgrade.

2. Discussion

We believe that batch time improvement can be achieved by upgrading the existing Hewlett Packard CPU-A and CPU-B. In order to achieve this goal, three options were considered:

* Option-A: Upgrade existing HP3000/950 to HP3000/980 and retain existing HP 3000/960
* Option-B: Upgrade HP 3000/950 and HP 3000/960 with a new HP 3000/992 CPU.
* Option-C: Retain the status quo situation.

The question to be answered is:

Which option would incur the least total cost within the next five year period?

Assumptions:

1. Prior probability of high system usage is expected to be .9. The prior probability of low system usage is .10.
2. For each option, costs are calculated for a period of five years using annual interest rates of eight percent.
3. Analysis assumes that the historical relationship among various variables at Meriter will continue into the future.

3. Statistical Data Collection and Analysis

In order to complete the quantitative analysis, statistical probabilities were calculated.

Representative sample was taken for this analysis. Given whatever information may be available to them regarding future system enhancements, day to day workload and their perceived need, they were asked to check one of the possible three categories for decision making for each type of system usage (high system usage or low system usage). Users were asked:

(1) If the system usage is high (high CPU utilization, slower response time), do they believe that their
a. Computation needs will increase by more than fifty percent, or,

b. Computation needs will remain same (change less than fifty percent up or down, or,
c. Computation needs will decrease by more than fifty percent.

(2) If the system usage is low (Low CPU utilization, faster response time), do they believe that their
a. Computation needs will increase by more than fifty percent. or,
b. Computation needs will remain same (change less than fifty percent up or down, or,
c. Computation needs will decrease by more than fifty percent.

Results of this experimentation are listed in Table One.

The responses indicated that if system usage is high, 90% believed that the new computation needs will increase, 8% believed that the new computation needs are will remain same and 2% believed that new needs will decrease.

When system usage was expected to be low, 30% believed that the new computation needs will increase, 10% believed that the new computation needs will remain same and 60% believed that new computation needs will decrease.

Based on the end user's conditional probabilities of software change requirements and a new system implementation for a given potential usage level, a joint probability Table was constructed (please see page 9, Table 1 and 2). The marginal probabilities of system usage (high or low) and the marginal probabilities of survey predictions (study suggesting upgrade, i.e. new computation needs will increase by fifty percent = X; study results are inconclusive, i.e. new computation needs will remain same (change less than fifty percent either direction = Y; and study suggests no upgrade, i.e. new computation needs will decrease by more than fifty percent = Z) were calculated. The posterior probabilities of given system usage (high or low) for given options were calculated.

In order to complete the quantitative analysis, we need the probabilities of various events. Shown below in table 1 are the estimates generated by Information Systems personnel. The estimates are based on past system experience together with the judgement of Information Systems management.

The probability of high system usage is expected to be .90, whereas the

probability associated with low system usage is .1. These probabilities are prior to any observation or experimentation.

Table 1: Conditional probabilities of software change requirements and new implementation for a given potential usage

EXPERIMENTAL RESULTS	POTENTIAL USAGE LEVEL; SYSTEM USAGE IS HIGH (H)	POTENTIAL USAGE LEVEL; SYSTEM USAGE IS LOW (L)
COMPUTATION NEEDS WILL INCREASE BY MORE THAN FIFTY PERCENT (X)	.90	.30
COMPUTATION NEEDS WILL REMAIN SAME - CHANGE BY LESS THAN FIFTY PERCENT UP OR DOWN (Y)	.08	.10
COMPUTATION NEEDS WILL DECREASE BY MORE THAN FIFTY PERCENT (Z)	.02	.60
TOTAL	1.0	1.0

The experimental probabilities listed above in table 1 will help evaluate the economic worth of the CPU upgrade. By using the original Information System personnel estimates (probabilities of high system usage=.9 and probabilities of low system usage=.1), and the experimental probabilities given in table 1, the joint probability table can be constructed.

Table 2: Joint Probability

POTENTIAL LEVEL OF SYSTEM USAGE	USAGE INCREASE BY MORE THAN 50% (X)	USAGE CHANGE BY LESS THAN FIFTY PERCENT UP OR DOWN (Y)	USAGE NEEDS DECREASE BY MORE THAN FIFTY PERCENT (Z)	EXPERIMENTAL RESULTS
HIGH SYSTEM USAGE	p(X AND H)=.81	p(Y AND H)=.072	p(Z AND H)=.018	p(H)=.9
LOW SYSTEM USAGE	p(X AND L)=.03	p(Y AND L)=.01	p(Z AND L)=.06	p(L)=.1
TOTAL	p(X)=.84	p(Y)=.082	P(Z)=.078	1.00

$$p(X \text{ AND } H) = p(X|H) * p(H) = .9 * .9 = .81$$
$$p(X \text{ AND } L) = p(X|L) * p(L) = .3 * .1 = .03$$

p(Y AND H) =p(Y|H) * p(H) = .08* .9 = .072
p(Y AND L) = p(Y|L) * p(L) = .1 * .1 = .01
p(Z AND H)= p(Z|H) * p(H) = .02* .9 = .018
p(Z AND L)= p(Z|L) * p(L) = .6 * .1 = .06

For each of the three options, high and low system usage probabilities are calculated. We cannot use the Information Systems personnel estimates (.9=high system usage and .1=low system usage) because those probabilities were estimated independently of experimentation. The posterior probabilities are now calculated (given on the next page). Although new system changes have not been made yet, the calculations are based on survey/experimentation.

p(H|X) = p(H and X)/p(X) = .81/.84 = .964
p(L|X) = p(L and X)/p(X) = .03/.84 = .036
p(H|Y) = p(H and Y)/p(Y) = .072/.082 = .878
p(L|Y) = p(L and Y)/p(Y) = .01/.082 = .122
p(H|Z) = p(H and Z)/p(Z) = .018/.078 = .231
p(L|Z) = p(L and Z)/p(Z) = .06/.078 = .769

All of the necessary information is now available and the expected monetary cost for each option can be calculated.

4. Financial Data and Analysis

Financial information for the respective options are calculated for five years. The information is available in Table 3 on next page. Each Option considers costs for both high and low system usage circumstances, including the following:

A. Yearly maintenance cost (net present value for 5 Years).
B. Present worth of annual HP maintenance costs.
C. Present worth of annual organization specific maintenance costs.
D. Multiple CPU vs single CPU advantages.
E. Lost strategic advantage cost, increased operational cost (if upgrade doesn't occur)
F. Price for each option (net cash out flow)

The total cost for each option was calculated. These costs were later used in the decision tree analysis as a total cost for a given option.

TABLE THREE – FIVE YEAR FINANCIAL COST CALCULATIONS

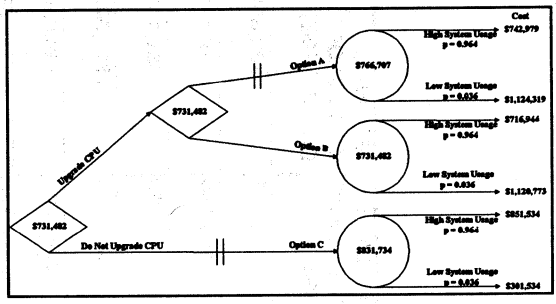
OPTIONS AND CATEGORY	OPTION–A HIGH USAGE (960/980)	OPTION–A LOW USAGE (960/980)	OPTION–B HIGH USAGE (992)	OPTION–B LOW USAGE (992)	OPTION–C HIGH USAGE (NO CHANGE)	OPTION–C LOW USAGE (NO CHANGE)
A. YEARLY MAINTENANCE COSTS	$32,352	$32,352	$22,752	$22,752	$32,352	$32,352
B. PRESENT WORTH OF YEARLY HEWLETT PACKARD MAINTENANCE COSTS	$129,182	$129,182	$90,849	$90,849	$129,182	$129,182
C. PRESENT WORTH OF YEARLY MERITER MAINTENANCE COSTS	$31,145	$31,145	$0	$0	$31,145	$31,145
D. TWO CPU ADVANTAGES	($10,000)	($10,000)	$10,000	$10,000	($10,000)	($10,000)
E. LOST STRATEGIC ADVANTAGE COST, INCREASED OPERATIONAL COST, ETC..	$0	$381,340	$0	$403,829	$668,555	$118,855
F. UPGRADE PRICE FOR EACH OPTION	$560,300	$560,300	$593,343	$593,343	$0	$0
G. TOTAL FIVE YEAR COST FOR EACH OPTION	$742,979	$1,124,319	$716,944	$1,120,773	$851,234	$301,534

ECONOMIC JUSTIFICATION FOR A SYSTEM UPGRADE

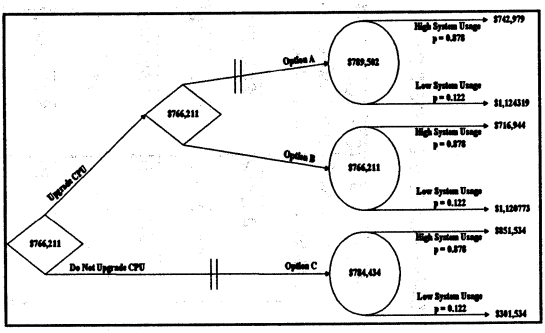
5. Decision Tree Analysis

A complete decision tree picture is given in the last page of this paper. Partial decision tree information is listed on the following pages with an explanation. Decision tree incorporates the probability information and the payoff or financial information to provide a solution or set of solutions.

Going from right to left and looking at the very top portion of the decision tree branch, we have two Options, Option A and Option B. Multiplying the respective system usage probability and corresponding costs for each Option, we find that Option B has a lower cost. We take the total cost of $731, 482 to the decision node. Similar analysis with Option C reveals a cost of $831,742. Going back further to our left we need to decide whether we want to upgrade the CPU or do not want to upgrade the CPU. Upgrading the CPU has a lower cost of $731,482. Thus we decide to choose Option B for this node.

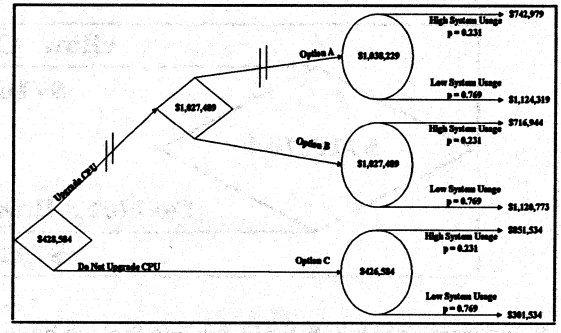


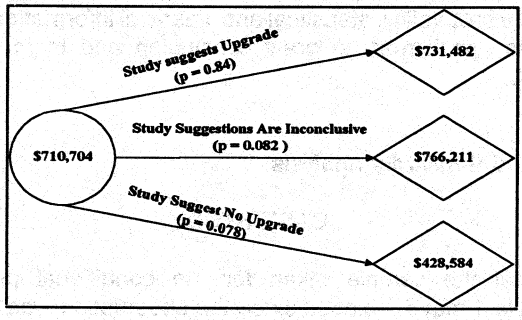
Going from right to left and looking at the middle portion of the decision tree branch, we have two Options, Option A and Option B. Multiplying the respective system usage probability and corresponding costs for each Option, we find that Option B has a lower cost. We take the costs of $766,211 to the decision node. Similar analysis with Option C reveals a cost of



$784,434. Going back further to our left we need to decide whether we want to upgrade the CPU or do not upgrade the CPU. Upgrading the CPU has a lower cost of $766,211. Thus we decide to choose Option B for this node.

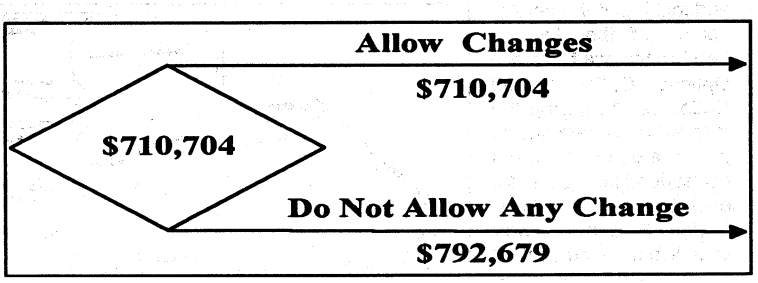
Going from right to left and looking at the bottom portion of the decision tree branch, we have two Options, Option A and Option B. Multiplying the respective system usage probability and corresponding costs for each Option, we find that Option B has a lower cost. We take the cost of $1,027,489 to the decision node. Similar analysis with Option C reveals a cost of $ 428,584. Going back further to our left we need to decide whether we want to upgrade the CPU or do not upgrade the CPU. Not upgrading the CPU has a lower cost of $428,584. Thus we decide to choose Option C for this node.





Going from right to left again, we find that we have three marginal probabilities. Probability that the new computation needs will increase is .84. Probability that the new computation needs will remain same is .082 and

the probability that the new computation needs will decrease is .078. From the earlier sections, the total cost is multiplied by the respective marginal probability. The resulting node has a cost of $710,704.



Going from right to left again, we find that we have two Options to allow any change or don't allow any change. The total costs $710,704 for allowing changes is lower than the total costs for not allowing any changes. Thus, we recommend that we allow changes.

Thus, based on the probability, statistical and financial information, decision tree analysis was performed to draw conclusion and to make further recommendations.

6. Probabilities and Sensitivity Analysis

CASE STUDY

It is possible that the sample taken for the conditional probabilities mentioned in Table 1 may be biased since the observation was performed by Information Systems personnel. Suppose the conditional probabilities of various events were different (i.e. instead of computation needs will increase; [p(X)] p=.9, the new p=.5; instead of computation needs will remain same [(p(Y)]=.08, the new p=.10; and instead of computation needs will decrease [p(Z)] probability=.02, the new p=.40). We want to know the resultant decision tree analysis decisions and the resultant cost totals. Table 4 and 5 lists conditional, joint and marginal probabilities.

Table 4: Conditional Probabilities of software change requirements and new implementation for a given potential usage level (Sensitivity analysis)

EXPERIMENTAL RESULTS	POTENTIAL LEVEL USAGE (SYSTEM USAGE IS HIGH)	POTENTIAL LEVEL USAGE (SYSTEM USAGE IS LOW)
COMPUTATION NEEDS WILL INCREASE BY MORE THAN FIFTY PERCENT (X)	.50	.30
COMPUTATION NEEDS WILL REMAIN SAME (CHANGES LESS THAN 50%) (Y)	.10	.10
COMPUTATION NEEDS WILL DECREASE BY MORE THAN FIFTY PERCENT (Z)	.40	.60
TOTAL	1.0	1.0

Table 5: Joint Probability

POTENTIAL LEVEL OF SYSTEM USAGE	EXPERIMENTAL RESULTS (X)	EXPERIMENTAL RESULTS (Y)	EXPERIMENTAL RESULTS (Z)	MARGINAL PROBABILITY OF SYSTEM USAGE
HIGH (H)	p(X AND H)=.45	p(Y AND H)=.09	p(Z AND H)=.36	p(H)=.9
LOW(L)	p(X AND L)=.03	p(Y AND L)=.01	p(Z AND L)=.06	p(L))=.1
MARGINAL PROBABILITY	p(X)=.48	p(Y)=.10	p(Z)=.42	TOTAL=1.00

p(X AND H) =p(X|H) * p(H) = .5 * .9 = .45
p(X AND L) =p(X|L) * p(L) = .3 * .1 = .03
p(Y AND H) =p(Y|H) * p(H) = .10* .9 = .09
p(Y AND L) =p(Y|L) * p(L) = .1 * .1 = .01
p(Z AND H) =p(Z|H) * p(H) = .40* .9 = .36
p(Z AND L) =p(Z|L) * p(L) = .6 * .1 = .06

For each of the three option (A, B and C), now we need to calculate, high and low system usage probabilities. We cannot use the information system personnel estimates (.9=high system usage and .1=low system usage) because those probabilities were estimated independent of experimentation. The posterior probabilities are now calculated (given on the next page). Although the new system changes have not been made yet, the calculations are based on sensitivity analysis experimentation.

p(H|X) = p(H and X)/p(X) = .45/.48 = .9375

$p(L|X) = p(L \text{ and } X)/p(X) = .03/.48 = .0625$
$p(H|Y) = p(H \text{ and } Y)/p(Y) = .09/.10 = .90$
$p(L|Y) = p(L \text{ and } Y)/p(Y) = .01/.1 = .1$
$p(H|Z) = p(H \text{ and } Z)/p(Z) = .36/.42 = .857$
$p(L|Z) = p(L \text{ and } Z)/p(Z) = .06/.42 = .142$

All of the necessary information is now available and the expected monetary cost for each options can be calculated. The calculated cost is given in table six on the following page.

The cost of not allowing changes was calculated to be $792, 679. The sensitivity analysis payoff results (listed in table six on page 16) reveal that despite dramatic probability change when the system usage is expected to be high, recommendations remain unchanged. i.e. When the study suggests that the computation needs will increase or when the study suggest that the computation needs will remain same, choose option B. When the study recommends that the computation needs will decrease, do not upgrade CPU.

The Sensitivity Analysis Payoff Results also reveal that despite dramatic probability change when the system usage is expected to be high, recommendations remain unchanged. However, the decision tree cost increases by $ 45,761 ($756,465-710,704). That is, when the study suggests that the computation needs will increase or when the study suggest that the computation needs will remain same, choose Option B. When the study suggest that the computation needs will decrease, do not upgrade CPU.

Additional cases with new probability could be constructed to review the effect of probability changes in decision making process.

7. Case Analysis - Cost Sensitivity Analysis

In the previous section, probability simulation was done to see the effect of conditional probability and the corresponding changes in the posterior probabilities and decision making process. Similar analysis can be performed for costs to calculate the sensitivity point where the costs for two decision nodes are equal. This will help assure the integrity of a solution recommendation.

TABLE SIX

OPTION / CATEGORY	OPTION–A HIGH USAGE	OPTION–A LOW USAGE	NODE TOTAL COST	OPTION–B HIGH USAGE	OPTION–B LOW USAGE	NODE TOTAL COST	OPTION–C HIGH USAGE	OPTION–C LOW USAGE	NODE TOTAL COST	LEAST EXPENSIVE NODE TOTAL	STUDY PROBABILITY
COST	742979	1124319		716944	1120773		851534	301534			
A. NEW COMPUTATION NEEDS WILL INCREASE	0.9375	0.0625	766813	0.9375	0.0625	742183	0.9375	0.0625	817159	742183	0.48
B. NEW COMPUTATION NEEDS WILL NOT CHANGE	0.9	0.1	781113	0.9	0.1	757327	0.9	0.1	796534	757327	0.1
C. NEW COMPUTATION NEEDS WILL DECREASE	0.857	0.142	796386	0.857	0.142	773571	0.857	0.142	772582	772582	0.42

EXPECTED COST = (.48*742183)+(.1*757327)+(.42*772582)= 756465

CASE ANALYSIS:

Table 3 provided the cost calculation for all three options with high and low system usages. Line G lists the total five year cost for each option. Whereas, all other costs are fairly straightforward, some doubts can be raised about lost strategic advantage costs, increased operational costs etc. for Option C given on line E. Table Seven on next page shows that if the lost strategic cost for Option C, when the system usage is high, is reduced from the original cost of $668,855 to the new cost of $563,697, it matches with the resulting node calculation recommendations of Option B when the study suggests an upgrade. The results are shown in Table seven on the following page.

What is the significance ?

What this really means is that the lost strategic advantage cost listed originally in Table three can go down from $668,855 to $563,697 for Option C before a decision change will take place. Between the range of $668,855 to $563,697 there is no change in recommendations.

8. Case Analysis - Recommendations

This study recommended the following for the Meriter Hospital.

* When the statistical study suggests that the computation needs will increase, Option-B (upgrading HP 3000/950 and HP 3000/960 with HP 3000/992) should be chosen.

* When the statistical study suggests that the computation needs will remain same, Option-B (upgrading HP 3000/950 and HP 3000/960 with HP 3000/992) should be chosen.

* When the study suggests that computation needs will decrease, Option-C should be chosen.

The observed probability and the given cost should be critically evaluated. The sensitivity analysis will help provide the ranges in which a given solution will remain unchanged.

TABLE SEVEN – REVISED FIVE YEAR FINANCIAL COST CALCULATIONS

OPTIONS AND CATEGORY	OPTION–A HIGH USAGE (960/980)	OPTION–A LOW USAGE (960/980)	OPTION–B HIGH USAGE (992)	OPTION–B LOW USAGE (992)	OPTION–C HIGH USAGE (NO CHANGE)	OPTION–C LOW USAGE (NO CHANGE)
A. YEARLY MAINTENANCE COSTS	$32,352	$32,352	$22,752	$22,752	$32,352	$32,352
B. PRESENT WORTH OF YEARLY HEWLETT PACKARD MAINTENANCE COSTS	$129,182	$129,182	$90,849	$90,849	$129,182	$129,182
C. PRESENT WORTH OF YEARLY MERITER MAINTENANCE COSTS	$31,145	$31,145	$0	$0	$31,145	$31,145
D. TWO CPU ADVANTAGES	($10,000)	($10,000)	$10,000	$10,000	($10,000)	($10,000)
E. LOST STRATEGIC ADVANTAGE COST, INCREASED OPERATIONAL COST, ETC..	$0	$381,340	$0	$403,829	$563,697	$118,855
F. UPGRADE PRICE FOR EACH OPTION	$560,300	$560,300	$593,343	$593,343	$0	$0
G. TOTAL FIVE YEAR COST FOR EACH OPTION	$742,979	$1,124,319	$716,944	$1,120,773	$746,376	$301,534

IV. CONCLUSIONS:

A systematic approach to the justification process is recommended. To recapitulate the concept, we, as Information Systems personnel, when convinced of a need to upgrade a system, could justify the upgrade by combining financial and statistical information into decision tree analysis for a better decision making process.

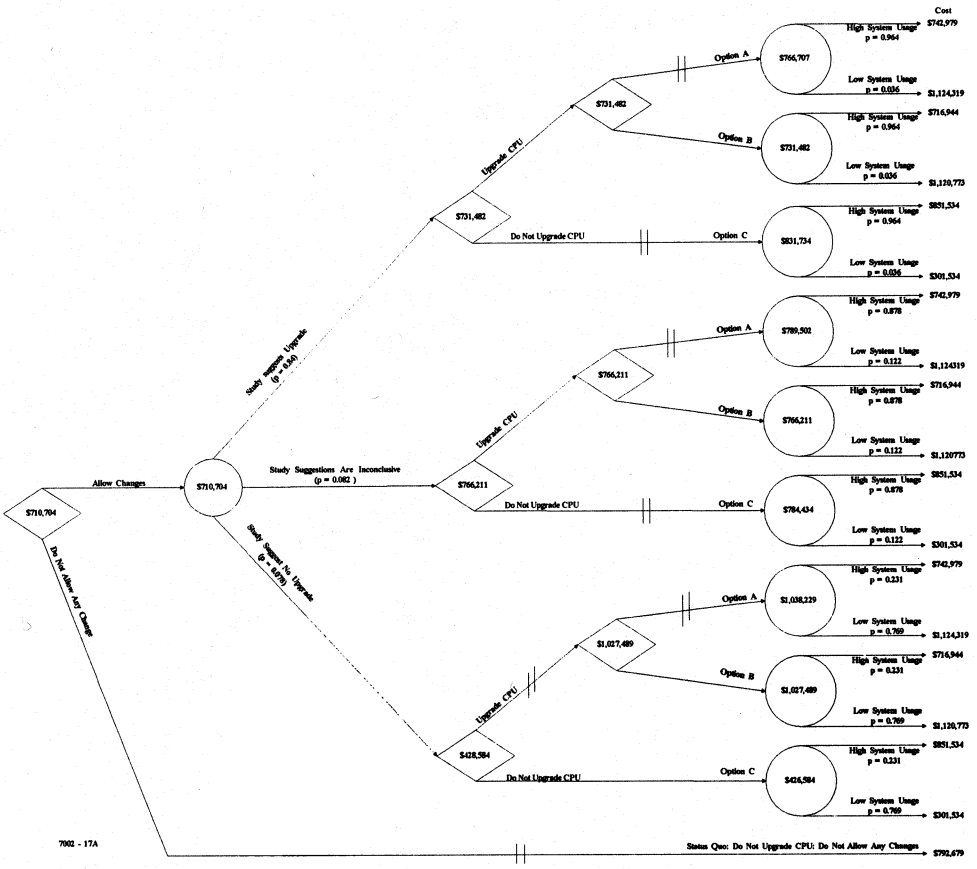
V. REFERENCES:

1. NEWMAN, DONALD G., 1988, Engineering Economic Analysis, San Jose, CA: Engineering Press, Inc.

2. CLARKE, BRUCE and DISNEY RALPH, 1985, Probability and Random Processes: a first course with application, New York, John Wiley and Sons

3. M. GUIGAN, JAMES R. and MOYER, R. CHARLES, 1989, Managerial Economics, New York, West Publishing Company

4. LUND, ROBERT, 1992, Interex paper # 3001, An overview of queue network modeling capacity planning, New Orleans proceedings


Cost
$710,704
Allow Changes
Do Not Allow Any Changes
$710,704
Study Suggests Upgrade (p = 0.50)
Study Suggests Inconclusive (p = 0.082)
Study Suggests No Upgrade (p = 0.070)
$731,482
Upgrade CPU
Do Not Upgrade CPU
$731,482
Option A
Option B
Option C
$766,707
High System Usage p = 0.964
Low System Usage p = 0.036
$742,979
$1,124,319
$731,482
High System Usage p = 0.964
Low System Usage p = 0.036
$716,944
$1,120,773
$831,734
High System Usage p = 0.964
Low System Usage p = 0.036
$851,534
$301,534
$766,211
Upgrade CPU
Do Not Upgrade CPU
$766,211
Option A
Option B
Option C
$789,502
High System Usage p = 0.878
Low System Usage p = 0.122
$742,979
$1,124319
$766,211
High System Usage p = 0.878
Low System Usage p = 0.122
$716,944
$1,120773
$784,434
High System Usage p = 0.878
Low System Usage p = 0.122
$851,534
$301,534
$428,584
Upgrade CPU
Do Not Upgrade CPU
$1,027,489
Option A
Option B
Option C
$1,038,229
High System Usage p = 0.231
Low System Usage p = 0.769
$742,979
$1,124,319
$1,027,489
High System Usage p = 0.231
Low System Usage p = 0.769
$716,944
$1,120,773
$426,584
High System Usage p = 0.231
Low System Usage p = 0.769
$851,534
$301,534
7002 - 17A
Status Quo: Do Not Upgrade CPU; Do Not Allow Any Changes
$792,679

SharePlex/iX: HP3000 Clustering Solution

By Bryan Dean
Hewlett-Packard Commercial Systems Division
19111 Pruneridge Ave, Cupertino, CA 95014
(408) 447-5591

Paper # 7003

The computer industry press and consultants have focused very heavily on clustering technology over the last several years. Even with all this press coverage, many CEO's, MIS managers, and MIS professionals are still asking themselves, why should I care about clusters? How will it help my business? And, how do I know if it will meet my business's needs? This paper's objective is to help answer some of these questions. Part 1 of this paper will give a brief definition and introduction to clusters. Part 2 will discuss five major points to consider when you are evaluating a cluster. Part 3 will analyze a fictitious mail order company's computing needs, and show how the HP3000 SharePlex/ix clustering implementation can meet those needs.

PART 1: An Introduction to Clusters

The term clustering was predominantly associated with Digital Equipment Corporation's VaxCluster line of solutions in the mid 1980's. Recently, many vendors have used the word clustering to describe their solutions and the term has developed a very broad definition. A high level definition of clustering is: Multiple systems loosely coupled through software, hardware and networking to provide computing capabilities that are unobtainable with a single system. Loosely coupled means that the systems are more closely linked than merely communicating with each other over networking, but are not as tightly integrated as a multi-processor based system like a HP3000 Series 992/200 where both of the processors are in one box.

Industry consultants have bounded this definition further by adding specific requirements. Aberdeen Group's 1992 research paper entitled, "Clustering: An alternative growth and operations path" offers this definition:

"A multi-node computer system which has
- o a single-system view as seen by users, programmers, operators and administrators
- o provisions for enhanced availability
- o system wide operations and management features
- o shared cluster-wide facilities for print queues, batch queues, file systems and peripherals
- o a graceful incremental growth capability
- o flexible configuration through interconnect and topology options."

What are the advantages of using multiple systems clustered together versus one large system? First, multiple systems allow data replication over a longer distance than a single system, so the data is protected from a disaster (fire, vandalism, natural disaster). A single system's data replication (like disk mirroring) capabilities are usually limited to a single data center, or at best a single building. If the whole data center was effected by a disaster (like the World Trade Center bomb explosion), both copies of the data would be lost. Even if the business had access to a system outside the range of the disaster, they would still have a slow recovery from off-site tapes, and all of the day's transactions would be lost.

Second, multiple redundant systems do not allow a single point of failure on a particular system to shut down the company's entire processing capabilities. Some fault tolerant systems also provide this redundancy, but the cost is very high, and loosing a single geographic location due to disaster remains an issue.

Third, clustering allows a modular horizontal growth path for adding new processing power or off-loading stressed systems during peak periods. If a mission critical application's performance goes down every time a query/quiz report is run, another system can easily be added. The query/quiz jobs are moved to the new system, and this frees up the main application. A single system implementation would require an upgrade, which would impact system availability, and might require an expensive box swap.

Forth, clustering distributed systems allows your computing environment mold around your business, rather than centering your business around your computers. The "glass house" environment often has difficulties being flexible, which results in slower application

development, and sluggish response to competitive pressures.

Most clustering solutions do not meet the Aberdeen Group definition, thus they don't provide all of the above clustering benefits. Most so called clusters recently introduced are only a glorified "SPU Switchover" solution. This usually entails two systems which can back up one another in the event that one of the systems fails. These systems usually both have physical connection to a single set of disk drives. There is certainly nothing wrong with this type of product because it serves a definite role; it is questionable, however, to term this a true cluster.

There are several "true" clusters on the market today that meet all or most of the Aberdeen Group definition. Each vendor has a different methodology of implementation, and each has its strengths and weaknesses. In evaluating a clustering solution, it is very important to concentrate on your business needs, not the technical implementation.

Part 2: Evaluating a Cluster

An MIS manager who is considering the move to clusters must contemplate the total impact to their business. Understanding the following five areas is critical:

1) How clustering impacts a company's business operations.
2) What effect the cluster has on the end user.
3) How a cluster impacts MIS operations.
4) The level and quality of vendor support
5) The cost of the complete clustering solution

1) Effect on Business Operations

A clustering implementation can give a business a distinct competitive advantage by providing a disaster tolerant computing environment, and by increasing the general data availability through reducing both planned and unplanned system outages. The business can also benefit through the flexibility that clustering brings.

If a disaster were to hit, all corporate eyes would

fall onto the MIS manager. One who has implemented a good recovery plan is an automatic hero... one who doesn't usually needs to polish the resume. A well implemented cluster solution should restore close to normal operations within 1 hour.

A cluster survives a disaster by making an exact copy (or replication) of the application environment, and making it available to the user. A disaster can hit and destroy the primary copy of the application, and the users would then simply access the secondary copy.

In evaluating a cluster for disaster tolerance, ask the following questions: Are there any distance limitations to the cluster's replication capabilities? Some disasters are considerate enough to limit themselves to a small geographic area, but others can effect a region for hundreds of miles. Can the entire application environment be replicated (databases, flat files, indexed files, programs, JCL, etc.), or can only the database be replicated? Data is definitely the most important asset to have replicated, but it is useless if the rest of the application environment is not available, or takes a long time for reconstruction.

Other replication questions to ask are: Is the replication automatic, or does it require operator intervention? Is the secondary data real time, or is it old and out of synch? Can the secondary data be accessed, or is it a stand-by copy only? Can one system be down without destroying the replication process, or can the secondaries "catch up" and re-synch themselves?

A clustering architecture that allows disaster tolerance gives the MIS manager a real opportunity to be a hero. Most businesses today are susceptible to natural disasters, vandalism, terrorism, data center fires, and other long term system outages. The effect on a business can be devastating. You might be asking yourself if you really need to be prepared for the rare disaster, but consider the following statistics taken from "Contingency Planning Strategies/90", 1992:

> 50% of all companies (reliant on computer technology) who experience a disaster and do not recover within 10 business days either never recover financially or file Chapter 11.
>
> A recent United States Government study has shown

that 93% of the firms which had a major data processing disaster were out of business in five years. The chances of surviving such an outage are only 7 in 100.

Industry research has shown that the critical functions of a business cannot continue more than 4.8 days after a catastrophe has occurred where no recovery procedures are in progress.

Most companies believe that their disaster recovery plan is sufficient, and that they have the expertise in house to recreate their computing environment. They fail to adequately estimate how long it will take to recreate the environment, and the resulting business impact due to the delay. The impact of the disaster, and in some cases the corporation's ability to survive, depends on how quickly they can re-establish processing operations. Again, consider the following statistics from "Contingency Planning Strategies/90", 1992.

The National Fire Protection Association estimates that over 40% of all companies which loose their vital records to a fire fail to survive the next business year.

The Average firm looses 2% to 3% of total gross sales within the first ten days after its data processing becomes non-operational.

This paper has focused heavily on disasters not because it is the most important feature of a cluster, but because it is the most often overlooked. This isn't a scare tactic, but an effort to cause significant thought on the subject. The following paragraphs will more briefly discuss the other positive effects that clustering has on the business.

The majority of system and data outages are not caused by disasters, but are attributed to the more mundane hardware, software and networking failures. A simple hardware failure that requires 6 hours of downtime (CE response, parts procurement, travel time, fix time, reboot time, etc.) can devastate certain industries, even if it only occurs once every two years. Imagine a car rental agency at a major airport suffering a 6 hour system downtime during rush hour business travel. The line at the counter would grow and grow, while the competitive agency at the very next counter would be speeding their clients into their cars. It wouldn't

take long before people in the long lines simply walked to another counter, possibly taking their loyal repeat business with them.

Lengthy system configurations, re-cabling, product installations and even data center moves can also cause serious down time. If you have complete "independent" redundancy, each of these scenarios can be handled by replicating the environment, switching the users to the secondary environment, and continuing operation. "Independent" redundancy means that the redundancy is not built into the same physical piece of equipment. Many fault tolerant systems have redundancy, but each piece can not be managed, changed or moved independently.

Look for a cluster that has the ability to fully replicate every component in the cluster. This allows you to choose the level of redundancy you are willing to pay for, and the level of risk you are willing to live with. Most companies are willing to live with some level of risk, but you don't want the cluster vendor to make that decision for you. You also want to examine the cluster's failover procedures. How long does it take to get the application back up and running again? Does it require operator intervention? You will pay a substantial premium for speed and automation. Be sure to carefully analyze what your business requirements are, and compare that to the cost.

High availability and disaster tolerance are not the only way that clustering impacts the business. The typical data center environment has traditionally required the business to structure itself around the data, rather than structuring the data such that it fits around the business. Clustering, as long as it is geographically unlimited, allows centralized or decentralized computing.

It is just as important to make sure the system management of the cluster has centralized and decentralized capabilities for maximum operational flexibility. You will need a solution that you can manage, or at least logically view, from a central location. Without this capability, you can imagine three operators in three different states (or countries) talking on the telephone with each other in order to synchronize their operations. The best solution to distributed, multi-system management is a

workstation or group of workstations that has complete monitoring and control of all the systems.

Another way clustering's flexibility benefits the business is in its modular growth path capability. You need to examine the scalability of the solution, as well as the top-end connectivity. Some clusters are limited to only two or three nodes, others can handle twenty, thirty, even a hundred or more nodes. Another key area to examine are the limitations of compatibility of nodes in the clusters. How many versions of the operating system can coexist? What type of networking connections are required to add a node? These questions will help you evaluate the true flexibility of the solution.

2) The Cluster's Effect on the End User

The goal of a clustering implementation should be complete transparency to the end user. An optimal solution should allow the user to view the entire cluster as one large system. Whether the user is running an application, querying multiple databases, or sending a print out to a remote printer, the user should not be required to know system I.D.'s or any networking commands.

The most difficult question to assess it what level of transparency is required in the event of a system failure. there are solutions that can completely hide failures within the cluster, and that have capabilities to redirect in-flight transactions. This level of failure usually requires re-engineering of the application, and a sophisticated two phase commit protocol transaction manager. The bad news is that this solution is as expensive as it sounds complicated (unless you are writing the application from scratch). Again, this is where you must decide if your business actually requires this level of implementation. Most business can live with a rare ten to thirty minutes system outage that requirs the users to re-establish their sessions on the secondary system. This is obviously not as desirable, but the level of complexity and cost is greatly reduced. In the future, the market will naturally demand shorter outages and seamless failovers as a standard.

Your users and applications should also see a neutral to positive performance impact with clusters. Some

times an application or group of applications seems better suited for a single large system environment. An example of this is some high disk I/O batch applications, or occasionally an online application that requires a tremendous amount of random physical disk I/O. Clustering somtimes struggles with this type of application distribution because all the disk I/O's must pass through networking, instead of directly between the CPU and its disc drives. In these cases, performance analysis can help determine the best way, if at all, to structure the application. Usually the typical end users using an OLTP application will not notice any performance difference between accessing local or remote data.

3) Effect on the MIS Operations department

Managing multiple systems which are loosely coupled and geographically distributed can be very challenging. Cost and complexity can get out of hand unless the cluster offers central monitoring and control capabilities. Monitoring the entire cluster using a standard console on each node of the cluster is very inefficient. A group of operators would be required to monitor many consoles, reading every message and waiting for some exception condition which requires their attention. A consolidated console implementation is a better solution, and the next step up from that is the concept of "management by exception" with the aid of a graphical user interface. This feature grows in importance as number of systems, the geographic distribution, and the general complexity of the cluster grows.

4) The Quality of vendor support for the cluster

If you decide that a clustering environment is a good fit for your business, the last thing you need is poor vendor support. First and foremost, the support should cover the entire cluster. The toughest problem you will encounter is the undefined problem where you do not know if it is networking, hardware, O/S, or the clustering software who's at fault. Unfortunately, undefined problems usually cause the customer to end up on the short of the stick because the various vendors involved all point at one another as the culprit. This is when you need a single telephone contact who will manage the problem to complete resolution. If multiple companies are involved, they must have a seamless approach to managing the support.

If you are looking for long term cluster support (which most people are), look for a vendor with a solid financial background, a good reputation and proven support capabilities. A vendor with an international support infrastructure should also be considered so you do not grow out of the vendor's capabilities. You never know when a merger, acquisition or multi-national expansion will add new requirements to the cluster support.

5) Cost of the solutions

All of the benefits and capabilities with clusters are useless if the solution is not cost effective. Be careful to note that the up-front cost is very rarely the bottom line. Make sure to understand the scalability of the solution. Some clusters today are very dependent on specialized, expensive, and inflexible add-on products. The optimal clustering solution allows easy conversion from a standard multi-node environment to a clustered environment, and an easy modular growth path. When moving from the "entry level cluster" to the "premium cluster", does it require new hardware, Networking, or Application code changes? If it does, you can be assured that you will pay a premium. These dependencies can lock you into a solution that will cost you heavily in the long run.

In summary, it is very easy to get caught up in the clustering phenomenon and the potential benefits to your business. Try to exercise a systematic approach to evaluating and analyzing the best solution. The most common mistake is over-buying on cluster capability bells and whistles that you and your business don't really need. The second most common mistake, however, is becoming too complacent with your MIS strategy and dismissing concepts like clustering which can lead to a uncompetitive business.

Part 3: Understanding SharePlex in a Sample Environment

Before diving into our sample environment, we will review the basics of the HP3000 SharePlex/iX solution. SharePlex is an umbrella architecture for a number of integrated products. The key enabling technologies are OpenView System Manager, which provides central management and control of the cluster, and NetBase, a set of software products which enables loose coupling of multiple HP3000 systems.

OpenView System Manager is a windows-based PC workstation product that works in conjunction with the HP3000, and utilize a very easy to use graphical user interface. The workstation can graphically show all of the clustered systems on a single screen, allowing every node in the cluster to be viewed by one operator. The workstation can also be divided up (up to seven workstations) so that particular tasks like tape management, I/O control, etc., can be automatically directed to a given workstation. This allows the MIS department flexibility in how to organize their operations. These workstations also have full console control over any system in the cluster, so an operatorless environment is possible for remote nodes.

Netbase is the second key technology in SharePlex. Where networking products (local or wide ares) physically connect the nodes within the cluster, Netbase can be considered as the "glue" that logically couples the nodes together. NetBase is a software product that resides between the applications and the operating system on each HP3000 node. This software intercepts application request on the operating system, and decides which node within the cluster owns the requested resource. Various caching and pre-fetching techniques are used by NetBase to greatly improve remote access performance.

NetBase is comprised of four fundamental components:

Network File Access (NFA), a central file system to provide file and database access transparently to users and application.

Shadowing, allows local or wide area network replication of data structures, programs, JCL, etc.

NBSpool Plus, a master print manager and router to give users and applications seamless access to any printer in the cluster

AutoRPM, which gives end users transparent access and execution capabilities to any program or application in the cluster.

Although OpenView and NetBase are the primary technologies, SharePlex leverages other strengths of the HP3000 software and related products. For example, AllBase SQL's features such as dynamic log switching, online backup, online database restructuring contribute

to the overall data availability. Products such as OpenView DTC manager allow single workstation management of all the Distributed Terminal Controllers. These are just two examples of the secondary products that serve specific functions within SharePlex.

For our sample environment, we will use a telephone based mail-order business called Widget TeleCat (or WTC). This example is not meant to show the only possible use of SharePlex, but merely an example of a very good business fit. WTC is roughly laid out as follows:

Site 1: Telemarketing Center in Miami, Florida

- o 500 Telemarketers answering telephones, inputting orders
- o Heavy OLTP application 2500 - 5000 orders input per hour
- o Ordering hours are 7 days/week @ 24 hours/day
- o Heavy read/updates on the Customer Database, heavy read and light update on the Product Database

Site 2: Shipping and Inventory Control Facility in Dallas, Texas

- o 24 hour shipping capability
- o Inventory control
- o Nightly batch updates
- o Heavy read/update on the Product Database, heavy read and light update on the Customer Database

WDT's typical business day (a very simplified example):

Telemarketing Center

7AM to 11PM - Telemarketers heavy transaction load
11PM to 7AM - Telemarketers light transaction load

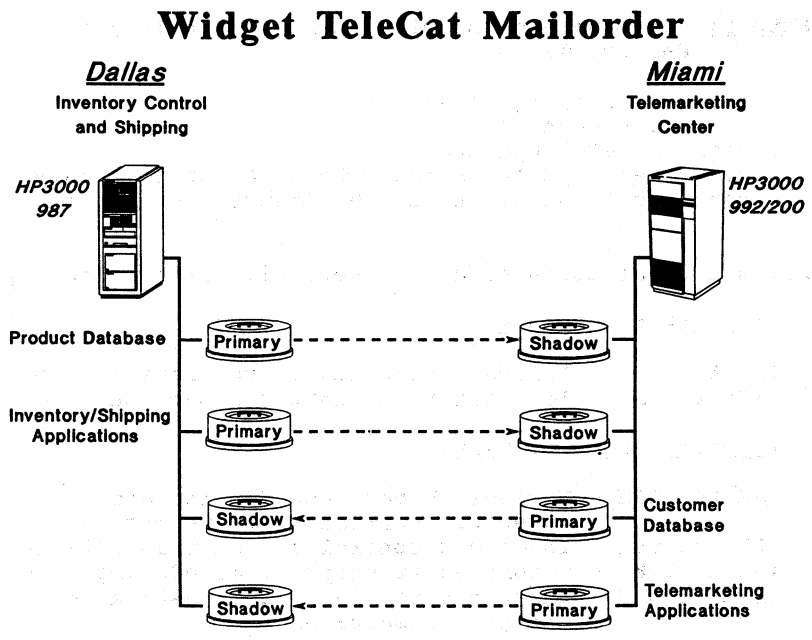
Inventory Control Center

7AM to 10AM - Various shipment scheduling and Inventory control query reports
10AM to 5PM - Inventory control online transactions
4PM to 7PM - Second shift shipment scheduling
11PM to 6AM - Inventory receiving and restocking online transactions

WDT's major MIS Priorities:

1) Order throughput of the 500 Telemarketers
2) 24 hour availability of the Telemarketing application
3) No system outage of over 30 minutes for any reason (including regional disaster)
4) Ease and cohesiveness of distributed systems management
5) Flexibility for future growth

As you can see (in diagram 1), the 992/200 in Miami shadows both the Customer Database and the complete Telemarketing application environment onto the Dallas 987. Likewise, the 987 in Dallas shadows the Product/Inventory Database and the complete Shipping/Inventory application environment to Miami.



(Diagram 1.)

By splitting up the application this way and utilizing the SharePlex configuration, all of the MIS related major issues are addressed. Let's take them one at a time.

1) Order throughput of the 500 Telemarketers.

Through offloading the query/quiz users to another node in the cluster, the Telemarketers have virtually exclusive use of the 992/200 so that telephone orders are completed quickly. Even if a series of CPU draining query/quiz reports were run utilizing the Product or Customer Database, they would only impact the 987 in Dallas. For the Customer Database updates that occur from the 987 based applications, NetBase would automatically direct the updates to the 992/200 with minimum impact. It is also important to note that this application division is completely transparent to the end user.

2) 24 hour availability of the Telemarketing application, and

3) No system outage of over 30 minutes for any reason

These two issue's go hand-in-hand in a SharePlex environment. SharePlex/iX is extremely well suited for disaster recovery through complete application environment replication. The replication capabilities includes databases (both Image and SQL), flat files, KSAM files, programs, JCL, etc. The replication function is automatically managed by the NetBase portion of SharePlex, and is virtually real time so the data at the secondary sites is completely up to date. This allows a full recovery of a company's business critical applications within 30 minutes. These replicated sites also have no distance limitations so that even the worst regional disaster can be withstood.

For example, if a fire were to break out in the Telemarketing data center, the telemarketers could simply log on to the secondary system in Dallas and continue their work. The major decision in planning and executing this strategy is determining how powerful a secondary system to employ. If the application is divided up well, the secondary system should be utilized heavily during normal operation, and have the capacity to handle all of the mission critical applications during a failover period.

The data redundancy also provides a "no single point of failure" configuration. Any component or group of components on one node can fail, and another complete application environment is available. SharePlex also provides the flexibility to incorporate single node high availability options without interfering with the multi-site redundancy. In this way, each node is as "bullet proof" as possible.

Long term planned downtime like major system re-configurations, O/S updates and even data center moves can also be handled through switching to the secondary site, performing all of the functions requiring downtime, and then switching back to the primary.

4) Ease and cohesiveness of distributed systems management

The OpenView System Manager provides SharePlex the multi-system management capabilities. The Miami office could have 2 window's based PC workstations to serve as the central cluster management. The two workstations could be used as follows:

Workstation 1: This workstation has the high level map of the systems, the Netbase functions, and the other major subsystems (as defined by operations). These functions are managed by exception, which means that the workstation informs the operators when there is an issue that requires operator attention. Instead of an operator constantly scanning the hundreds of messages rolling off the screen, they are free to perform other functions. For example, if shadowing went down between Miami and Dallas, the system icon (graphical symbol on the screen) or the NetBase icon would turn red to alert the operators of the event. OpenView could also be set up to page the operators or technical support staff if they were away from the workstation. All events requiring operator attention can be monitored by this system map workstation.

Workstation 2: This workstation is more dedicated to the control of the cluster. Through windows, the operator can bring up any system in the cluster and perform any command or action required. For example, the operator could restart shadowing, respond to tape requests or even perform system start-up and shut-down.

OpenView System Manager can also be configured for

automated response which is useful for the many system events that require a standard response every time they occur. If we continue on our shadowing example, the system could automatically restart the shadowing function without any delay or operator intervention.

5) Flexibility for future growth

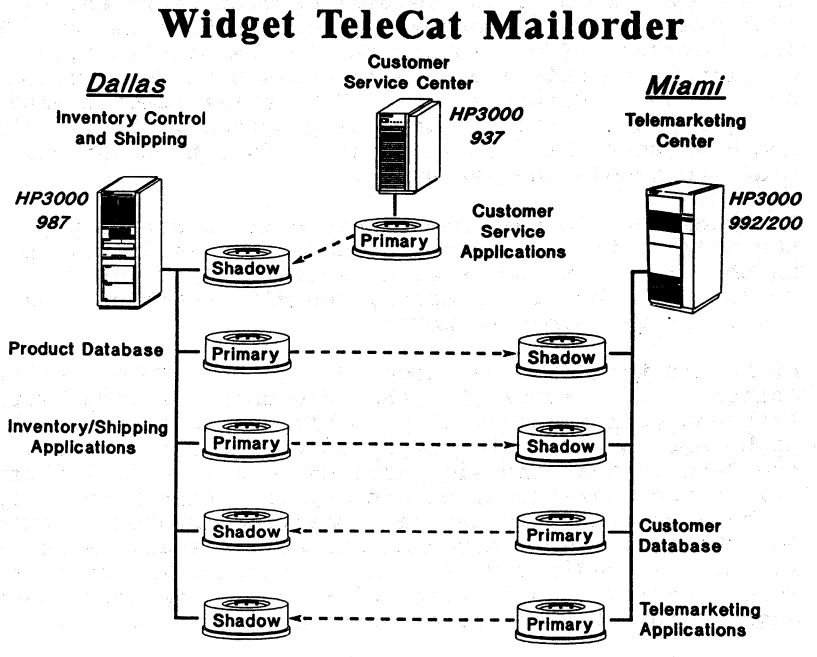
Let's suppose that Widget TeleCat decides they want to add a customer service department, and so WTC has secured a building in Miami across town from the Telemarketing Center. The MIS department has decided that the group of 30 customer service representatives will use a very impressive application that is very CPU and screen I/O intensive. The application is completely interactive, and accesses both the Customer and Product Database, yet MIS still doesn't want a heavy impact on the telemarketing systems. It is also decided that MIS doesn't want a full time operations staff to handle the new system.

The solution to this new scenario is to either upgrade the 992/200 to a 992/3000, or add a 937 as a new node in the cluster to serve as a dedicated Customer Service system. If the 937 is added, the MIS department has a couple of decisions to make:

A) Should the 937 be located in the Telemarketing Center, or local to the Customer Service Reps? Although performance will probably not be an issue, the high screen I/O rates might be better suited local to the Reps so that the wide area network load is lighter. All of the operations, except physical tape mounts, can be handled centrally by the OpenView System Manager workstations, so increased operations staff is not an issue. A dedicated 937 also guarantees that the Telemarketing application will not have performance degradation if there is a run on customer service.

B) Should the databases be shadowed to the Services Center, or should the applications utilize the central file system (NFA) to access the databases? All of the application code would be local, and either method will be completely transparent to the Reps and their application. The shadowing option will provide better read performance of the application, and will provide an extra level of redundancy. The NFA option is cheaper, however, and is a little easier to manage.

To complete our example, Widget TeleCat will choose NFA and utilize the Customer Service Center 937 as a client who uses the Miami and Dallas systems as the back-end data server. As you can see in diagram 2, the Customer Service Center system only locally houses the application software. Our assumption here is that the business is comfortable with their current level of database redundancy, and that since the interactive application is not disk I/O bound on the databases, the Customer Service Reps will not notice any performance degradation.



(Diagram 2.)

The preceding example has hopefully shed some light on how SharePlex can be applied to a business's computing needs. Many other environments will also fit into a coupled systems solution, and you will see more and more vendors throwing their hats into the clustering arena. The computing trend is obviously moving away from very large monolithic glass houses to the more

agile multi-system variety. And faster networking speeds, client-server computing and heightened awareness of disaster recovery are but a few of the variables that are shaping the outcome. Today, there isn't any one solution that is the panacea of all computing challenges, but there are several excellent solutions that meet today's specific business needs.

Paper Number: 7004

Printing Checks on LaserJets

Debra B. M. Canfield
Dairylea Cooperative Inc.
P.O. Box 4844 Syracuse, New York 13221-4844
315-433-0100

We've been printing checks on LaserJet printers at Dairylea for about a year and a half. Prior to that time we used a 900 line per minute HP2566 line printer. To say that we're happy about the change would be an understatement. We're thrilled, but as I talk with people about how checks are being printed at other companies, I find that most people are still using line printers.

When I think about why this is probably so, it reminds me of the case of the Christmas ham. Once upon a time there was a newly married couple who went out to buy a ham for their first Christmas together. They picked out a nice one, and the wife asked the butcher to cut off a few inches on the small end and wrap them up together. Her husband asked her why she was doing that. She said. "That's just what you do with hams, and it's certainly a lot easier to have the butcher cut off the end than to do it when I get home." "But, why cut off the end at all?" he wanted to know. She replied that her mother always did it that way. Well, he wasn't convinced. *His* mother never cut the ends off hams. He asked his wife to check with her mother and find out why. "Why?" her mother replied, "because if I didn't cut off the end, it wouldn't fit in my pan." The new husband smiled when he heard the reason and said they would buy a larger pan.

Why do we keep printing checks on line printers when we can print them on LaserJets? The biggest reason is probably because we've always done it that way, and we don't have the time or knowledge to work out a better alternative.

In this paper I'll begin by discussing the benefits of printing checks on LaserJets. I expect that anyone reading this paper believes that there are benefits to be gained, and probably already knows most of them. For this reason, I'll go over the benefits rather quickly. I am including them in case you may have missed some, or need a list to use as support to present a case to make the change from printing checks on line printers to LaserJets.

After talking about the benefits, I'll describe the basic parts of a check, so you can understand exactly what you need to print to create a check. Then I'll devote the bulk of the paper to the practical considerations of how to go about creating them on LaserJets. Finally, I'll look at some security issues. At Dairylea we use an HP3000, but most of the concepts are the same whether you use an HP3000, a Unix computer, or even a personal computer.

Benefits

Printing checks on LaserJets is easy.

Easy to Look At

Checks printed on LaserJets are easier to look at than checks printed on line printers. The overall appearance of a LaserJet check is more professional, and LaserJets offer more flexibility than line printers. Line printers just can't compete with the wide variety of fonts and the 300 or 600 dots per inch (dpi) resolution available on LaserJets.

Illustration 1 on page 3 shows an example of our old line printer checks along side our new LaserJet checks. Notice, for example, the differences in fonts and table arrangements, how they contribute to the overall appearance of the check and how they draw attention to the most important information on the stub.

Good line printers provide some flexibility. For example, you can print at six or eight lines per inch, and if your line printer has more than one size font, you can change the number of characters you can print per line.

With LaserJets you can vary your line spacing by any number of dots; for a standard LaserJet, that's by the 300ths of an inch. The number of characters per line depends on the font selected, but since there are so many more fonts available for LaserJets, you have much more flexibility. With LaserJets you can print lines, boxes and graphics wherever you want them.

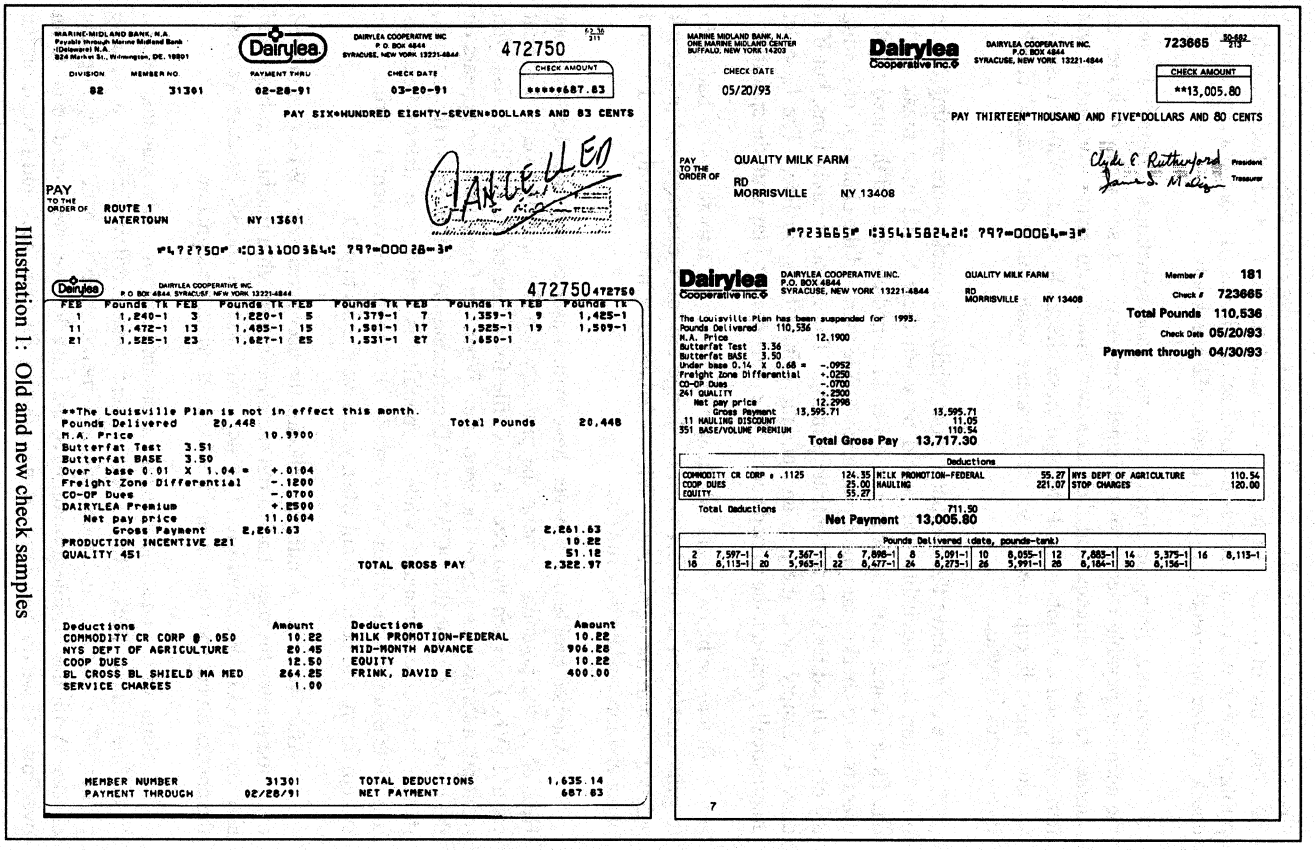
Easy to Produce

One of the things I appreciate as the manager of an operatorless computer environment is how much easier it is to print checks on LaserJets than it was with the line printer. At Dairylea we print a number of different kinds of checks, accounts payable, milk, hauling, third-party, etc., for a number of different companies. The following table compares what we had to do to print checks on the line printer with what we have to do now with a LaserJet.

Line Printer Checks	LaserJet Checks
Find the checks and verify starting number.	
Load the checks in the printer and line them up.	Go in the computer room and take the checks off the printer.
Watch for check breaks and printer jams.	

With preprinted checks on the line printer we had to either enter the check number in response to a console prompt or examine the spoolfile to verify the

Printing Checks on LaserJets 7004-3



MARINE-MIDLAND BANK, N.A.
Payable through Marine Midland Bank
(Delaware) N.A.
824 Market St., Wilmington, DE. 19801

Dairylea

DAIRYLEA COOPERATIVE INC
P O BOX 4844
SYRACUSE, NEW YORK 13221-4844

472750

DIVISION	MEMBER NO	PAYMENT THRU	CHECK DATE	CHECK AMOUNT
02	31301	02-28-91	03-20-91	*****687.83

PAY SIX*HUNDRED EIGHTY-SEVEN*DOLLARS AND 83 CENTS

PAY TO THE ORDER OF ROUTE 1
WATERTOWN NY 13601

⑈472750⑈ ⑆031100364⑆ 797-00028-3⑈

Dairylea — DAIRYLEA COOPERATIVE INC. P.O. BOX 4844 SYRACUSE, NEW YORK 13221-4844 — 472750 472750

FEB	Pounds Tk	FEB	Pounds Tk	FEB	Pounds Tk	FEB	Pounds Tk	FEB	Pounds Tk
1	1,240-1	3	1,220-1	5	1,379-1	7	1,359-1	9	1,425-1
11	1,472-1	13	1,485-1	15	1,501-1	17	1,525-1	19	1,509-1
21	1,525-1	23	1,627-1	25	1,531-1	27	1,650-1		

**The Louisville Plan is not in effect this month.

Pounds Delivered	20,448	Total Pounds	20,448
M.A. Price	10.9900		
Butterfat Test 3.51			
Butterfat BASE 3.50			
Over base 0.01 X 1.04 =	+.0104		
Freight Zone Differential	-.1200		
CO-OP Dues	-.0700		
DAIRYLEA Premium	+.2500		
Net pay price	11.0604		
Gross Payment	2,261.63		2,261.63
PRODUCTION INCENTIVE 221			10.22
QUALITY 451			51.12
	TOTAL GROSS PAY		2,322.97

Deductions	Amount	Deductions	Amount
COMMODITY CR CORP @ .050	10.22	MILK PROMOTION-FEDERAL	10.22
NYS DEPT OF AGRICULTURE	20.45	MID-MONTH ADVANCE	904.26
COOP DUES	12.50	EQUITY	10.22
BL CROSS BL SHIELD MA MED	264.25	FRINK, DAVID E	400.00
SERVICE CHARGES	1.00		

MEMBER NUMBER	31301	TOTAL DEDUCTIONS	1,635.14
PAYMENT THROUGH	02/28/91	NET PAYMENT	687.83

MARINE MIDLAND BANK, N.A.
ONE MARINE MIDLAND CENTER
BUFFALO, NEW YORK 14203

Dairylea Cooperative Inc.

DAIRYLEA COOPERATIVE INC.
P.O. BOX 4844
SYRACUSE, NEW YORK 13221-4844

723665 50-662 213

CHECK DATE 05/20/93

CHECK AMOUNT **13,005.80

PAY THIRTEEN*THOUSAND AND FIVE*DOLLARS AND 80 CENTS

PAY TO THE ORDER OF QUALITY MILK FARM
RD
MORRISVILLE NY 13408

Clyde C. Rutherford President
James S. Maling Treasurer

⑈723665⑈ ⑆354158242⑆ 797-00064-3⑈

Dairylea Cooperative Inc. — DAIRYLEA COOPERATIVE INC. P.O. BOX 4844 SYRACUSE, NEW YORK 13221-4844 — QUALITY MILK FARM RD MORRISVILLE NY 13408

Member # 181
Check # 723665
Total Pounds 110,536
Check Date 05/20/93
Payment through 04/30/93

The Louisville Plan has been suspended for 1993.

Pounds Delivered	110,536		
M.A. Price	12.1900		
Butterfat Test 3.36			
Butterfat BASE 3.50			
Under base 0.14 X 0.68 =	-.0952		
Freight Zone Differential	+.0250		
CO-OP Dues	-.0700		
241 QUALITY	+.2500		
Net pay price	12.2998		
Gross Payment	13,595.71	13,595.71	
11 HAULING DISCOUNT		11.05	
351 BASE/VOLUME PREMIUM		110.54	
	Total Gross Pay	13,717.30	

Deductions

COMMODITY CR CORP @ .1125	124.35	MILK PROMOTION-FEDERAL	55.27	NYS DEPT OF AGRICULTURE	110.54
COOP DUES	25.00	HAULING	221.07	STOP CHARGES	120.00
EQUITY	55.27				

Total Deductions 711.50

Net Payment 13,005.80

Pounds Delivered (date, pounds-tank)

2	7,597-1	4	7,367-1	6	7,898-1	8	5,091-1	10	6,055-1	12	7,883-1	14	5,375-1	16	8,113-1
18	8,113-1	20	5,965-1	22	8,477-1	24	8,275-1	26	5,991-1	28	8,184-1	30	8,156-1		

7

Illustration 1: Old and new check samples

check number entered by the user. When we used the same check stock for more than one kind of check, we had to wait until the first set of checks finished printing before we could process the next set of checks. If someone entered the check number incorrectly, they had to rerun the check job.

Sometimes lining up checks went well, and sometimes it didn't. I hated the accounts payable checks for one small company. Those checks took me ten minutes to do the lineup, and then took two minutes to print. At times I was tempted to tell accounts payable to type those checks by hand. With a LaserJet, we don't need lineup checks, so we eliminate wasted time and wasted checks. Every check is perfect.

We *could* leave the computer room while the checks were printing on the line printer, but inevitably that would be when there would be a break in the checks or a paper jam. Paper jams resulted in ruined checks and the need for the users to void checks and type replacements. We then had the challenge of getting the checks lined back up to continue printing. The LaserJet rarely jams, but if it does we simply clear the jam and put the printer back on line. The LaserJet automatically reprints the ruined check. We don't need to play with the spoolfile.

Before we had a LaserJet dedicated to checks, the only things we had to do to print checks on a LaserJet were to change the toner cartridge and put check safety paper in the paper tray. With a dedicated printer, we only need to add paper and toner when they run out.

Another problem we never have with LaserJets that we did with our HP2566 is the potential to hang the printer when downloading vertical format control (VFC). When you need a VFC file for a line printer in this class, you must put a forms message on the spoolfile. When the forms message appears on the console, you issue the DOWNLOAD command. As HP warns in the manual, if you issue the command while a spoolfile is active, the device becomes unavailable until you restart the system. It's never convenient to bring your system down, and certainly not when you have checks to print.

After we finish printing the checks, the user takes over. When we printed on a line printer the users had to run the checks through a machine to burst and sign the checks. We eliminated this step with LaserJet checks, because we print the checks on cut sheet paper, and the LaserJet prints the signature.

Some of our check jobs require more information on the stub than would fit on line printer checks. In the past we sent these overflow sheets to a separate printer. The users would then manually merge them in with the appropriate checks. With the flexible formatting of the LaserJet, we rarely need to go to an overflow. If we do, we simply turn on duplex for that check and print it on the

back, or take a plain piece of paper from the other tray and print it right after the check, keeping everything in the proper order.

For members who elect payment via direct deposit, we similarly take a piece of paper from the plain paper bin and print a confirmation that looks almost like a check. We benefit by having the checks all in member order, whether they receive a check or a direct deposit confirmation. Besides the benefit in ease of handling prior to mailing, the microfiche copies of the checks and direct deposit confirmations we produce are now all in member number order, rather than being sorted first by whether they are checks or direct deposits.

Easy to Change

With LaserJets, it's easy to make both mandatory and cosmetic changes. Sometimes you *have* to change your check format. For example, a few months ago our bank notified us that they were changing the bank through which they clear our checks. Therefore, we had to change the ABA transit numbers on our checks. With preprinted check stock, we would have had to destroy the old stock and order new. Besides the money lost on the unusable check stock, it would have taken weeks to get new checks. Since we print our checks on LaserJets, we could easily make the necessary changes and implement them on the appropriate day. We had no wasted stock and no delay.

Other changes you want to make may not be absolutely necessary but will improve the appearance of your checks. You probably wouldn't throw out your old check stock, but would make the change when it was time to order new checks. With LaserJet checks, you don't need to wait. For example, the preprinted checks we used for one company didn't include their logo. When we were making the move to printing on LaserJets, I asked them whether they would like to have their logos on their checks along with their company name. That little addition earned us positive points with little effort. It's no problem for us if they redesign their logo, or change their address.

Easy on the Budget

Printing checks on LaserJets can save you money. Our preprinted **check stock** costs ran from $47 to $155 per thousand, depending on the check. Some checks required a special size envelope, making the total cost even higher. The plain check safety paper we now use for all our checks costs $28 per thousand.

Printing checks on LaserJets reduces **inventory carrying costs** by eliminating the need to keep an inventory of many different checks. Using blank check safety paper, we only need to store one type of check stock. Eliminating all the special check stock also makes our disaster recovery plan easier. We don't need to store all those different kinds of checks off site.

Format changes are easy to make with LaserJets. They save money by eliminating the loss from destroying checks when you must make changes, and they eliminate printer plate charges to make changes on preprinted forms.

You can save on **printer cost and maintenance.** In our case we replaced an HP2566 with a LaserJet IIIsi, and we save $170 every month on maintenance. We sold the HP2566 for $4,100 and paid $4,500 for the LaserJet IIIsi.

It's not easy to find a change that produces better results with less human effort, and at a lower cost, but moving from printing checks on line printers to printing them on LaserJets can do all of that. Perhaps unanswered questions are the reason more companies haven't made the change. That's what I'll try to cover in the rest of this paper, with the objective of providing the information you need to get started with LaserJet checks and the answers to questions people may ask.

The Parts of a Check

Illustration 2 on page 7 shows an example of a check with areas marked corresponding to the description that follows.

1. **Bank information** includes the bank name and location, check number, the ABA transit number in fraction form and the MICR coding along the bottom. The MICR coding includes the check number, ABA transit number and account number.

2. **Company information** includes the company logo (optional), name and address and authorized signatures.

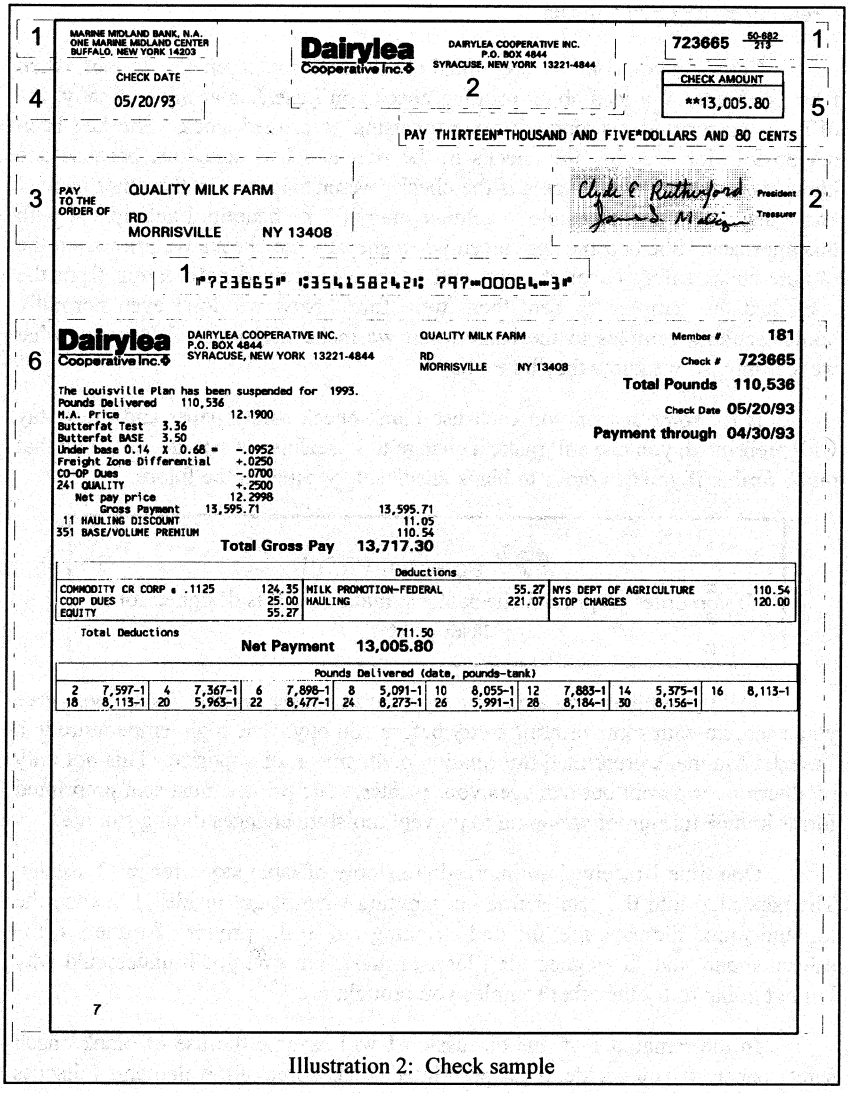
3. **Payee information** includes the payee's name and optional address.

4. The **check date** is the date the check becomes cashable. It need not be the date of printing.

5. You don't need to write the **amount** of the check out in words, but it is a good idea to do so, because that makes it more difficult for someone to change the amount. The check programs we write include the amount in words. Our accounts payable purchased software only shows the amount as digits.

6. The existence and layout of the **check stub** are entirely dependent upon your needs. Most business applications use a stub to explain the check.

Will the increase of direct deposit eventually make checks obsolete? If so, and if it may be soon, why bother to make changes to check programs? At Dairylea we have offered direct deposit to our members for a number of years. Over time more have chosen direct deposit, but this has in no way eliminated the



MARINE MIDLAND BANK, N.A.
ONE MARINE MIDLAND CENTER
BUFFALO, NEW YORK 14203

CHECK DATE
05/20/93

Dairylea Cooperative Inc.

DAIRYLEA COOPERATIVE INC.
P.O. BOX 4844
SYRACUSE, NEW YORK 13221-4844

723665 50-682 213

CHECK AMOUNT
**13,005.80

PAY THIRTEEN*THOUSAND AND FIVE*DOLLARS AND 80 CENTS

PAY TO THE ORDER OF
QUALITY MILK FARM
RD
MORRISVILLE NY 13408

Clyde C Rutherford President
James M Madigan Treasurer

⑈723665⑈ ⑆354158242⑆ 797-00064-3⑈

Dairylea Cooperative Inc.

DAIRYLEA COOPERATIVE INC.
P.O. BOX 4844
SYRACUSE, NEW YORK 13221-4844

QUALITY MILK FARM
RD
MORRISVILLE NY 13408

Member # 181
Check # 723665
Total Pounds 110,536
Check Date 05/20/93
Payment through 04/30/93

The Louisville Plan has been suspended for 1993.

Pounds Delivered	110,536	
M.A. Price	12.1900	
Butterfat Test	3.36	
Butterfat BASE	3.50	
Under base 0.14 X 0.68 =		-.0952
Freight Zone Differential		+.0250
CO-OP Dues		-.0700
241 QUALITY		+.2500
Net pay price	12.2998	
Gross Payment	13,595.71	13,595.71
11 HAULING DISCOUNT		11.05
351 BASE/VOLUME PREMIUM		110.54
Total Gross Pay		**13,717.30**

Deductions

COMMODITY CR CORP @ .1125	124.35	MILK PROMOTION-FEDERAL	55.27	NYS DEPT OF AGRICULTURE	110.54
COOP DUES	25.00	HAULING	221.07	STOP CHARGES	120.00
EQUITY	55.27				

Total Deductions 711.50

Net Payment 13,005.80

Pounds Delivered (date, pounds-tank)

2	7,597-1	4	7,367-1	6	7,898-1	8	5,091-1	10	8,055-1	12	7,883-1	14	5,375-1	16	8,113-1
18	8,113-1	20	5,963-1	22	8,477-1	24	8,273-1	26	5,991-1	28	8,184-1	30	8,156-1		

Illustration 2: Check sample

need for our check programs. We still need to calculate the checks and send a report to our members. Using the same layout as an actual check, minus the bank information and signatures, gives the members a report they understand. When a member elects direct deposit, users set a master file flag, and when the member comes up for direct deposit, the printer simply grabs a piece of plain paper from the top tray instead of a piece of check safety paper from the bottom tray.

Creating Your Own Checks

Certainly you *can* buy preprinted check stock for LaserJets. In fact, when I first presented my idea about printing checks on LaserJets at my company, the office manager was a strong advocate for using preprinted stock. She had been responsible for ordering the checks in the past and had concerns, because she knew there would be problems if the checks weren't right. On the other hand, I knew that we wouldn't be able to achieve many of the benefits I anticipated with this approach. She became convinced when she saw the checks we printed on the sample check safety paper she got, and when we received notification from the bank that the samples we sent them were fine. Now we don't even normally bother sending samples to the bank when we make a change. We know what we're doing, so we know they'll be fine.

If for some reason you can't use blank check safety paper and must stay with preprinted, you can still make a change to LaserJets. If you do, you'll be that much further if you can move to blank check safety paper in the future.

Potential Pitfall

If you order preprinted check stock, make sure it is designed for a laser printer.

Tell the supplier you are going to use it in a laser printer. Make them give you a sample with some printing to try before you buy. The high temperature in a LaserJet can make preprinted information melt, smear, or vaporize. This not only produces poor output but damages your printer. The printer must seal preprinted forms in moisture-proof wrapping to prevent moisture changes during storage.

One time I received an interesting sample of label stock for my LaserJet. The paper fed into the printer fine but jammed when it got inside. I opened the top and found the page melting and stringing out in the printer. I turned to the salesman and said, "Designed for a laser printer? I'm sure you'll understand why I'm not going to try the other samples you brought me."

In the remainder of this discussion I will assume the use of blank check safety paper. If you decide to use preprinted stock, some of the elements I discuss will be preprinted for you.

The Key Ingredients

Check Safety Paper

Most companies use check safety paper for printing checks. The paper is available in a variety of colors and has a finely patterned background to make it

difficult to copy or modify the check. Our bank's requirements are that the paper must be 24 lb. MICR bond. Check with your own bank for their requirements. Check safety paper is available from many suppliers. The best place to start is probably with the company that currently supplies your preprinted checks.

MICR Toner and Fonts

MICR stands for magnetic ink character recognition. It's the technology most banks use to read the line of numbers printed on the bottom of the check in a special MICR font. The font has characteristics the same way that Courier or Times Roman or any other font does. Unlike some fonts, it doesn't have special effects like bold or italic, and there is only one size. Only the line of MICR numbers *must* be printed with MICR toner, but you can print the whole check with MICR toner, which is what you do when you print the whole check from scratch on a LaserJet.

There are a number of companies that sell MICR toner for LaserJets. Hewlett-Packard sells MICR toner only for the series II, III and IIID. We buy our MICR toner for the LaserJet IIIsi from MICRTech (address under references at end of paper).

The MICR font is available from HP in a font cartridge and undoubtedly from other font cartridge vendors as well. MICR is also available as a soft font from a number of suppliers. We use the MICR soft font included with Fantasia from Proactive Systems, since we use their software to produce our checks (address under references at end of paper). Fantasia automatically keeps track of whether the MICR font is in the printer and downloads it when necessary. Warning! If you use a MICR soft font on a LaserJet series 4, you must adjust the horizontal motion index to 8 characters per inch just after calling the soft font and just before printing the MICR line. You do not need to do this with earlier models of the LaserJet.

Potential Pitfall

Make sure you place the MICR coding exactly.

There is little margin for error. Because of this, even though the bank part of the numbers remains the same and we could include it with the basic check form, since the check number part varies, we wait and print the line all at once after we determine the check number. This way we do not risk bad spacing between the bank number and the check number. We always go to exactly the same place to print the MICR information. We have a little plastic MICR document template we lay over the check to verify the exact placement and have never had a sample check rejected by the bank.

The ABA transit number portion of the MICR information must be in exactly the same spot for all checks. The order, layout and length of other MICR number elements vary between banks and accounts. Special symbols mark the beginning and end of each section for most banks. You can see these elements by looking at your preprinted checks, but you should contact your bank to get a specification sheet for your checks.

Including Logos and Signatures

One of the things we like about producing our own checks is how easy it is to change logos and signatures. We no longer need to order new checks when a logo changes or buy a new signature plate for a check signing machine. We can make all the changes ourselves.

To include logos and signatures on LaserJet checks, you need to accomplish three tasks.

1. Get the logo or signature into a computer understandable form.
2. Turn the computer code into PCL (printer command language).
3. Include the PCL in the right spot on your check.

You can accomplish the first step by scanning the logo or signature. We use our ScanJet scanner and Scanning Gallery software to create a TIFF file. We set the resolution to 300 dots per inch and make the scan area as close to the image as possible, because any white space becomes part of the image and makes final placement on documents more difficult. For checks with two signatures, we scan each signature separately in case we need to use one of the signatures by itself or in combination with a different signature on another document.

For optimum print quality, scan an original that is the same size as what you want to print. For example, we print Dairylea logos in both one and two inch sizes. Rather than scanning one logo and scaling it to both sizes, we scanned original logos in each size.

The way to accomplish the second and third steps depends on whether you do all your own PCL coding or whether you decide to use a tool like Fantasia. Either way, you can begin by using a word processor to print the logo or signature to a file using a LaserJet printer driver and specifying a resolution of 300 dots per inch. We use Microsoft Word, left justify the graphic and set all margins to zero before printing. Because a word processor created this file, it contains general commands at the beginning and end for page orientation, line spacing, fonts, etc. You must remove these commands to create a general graphics raster file to print anywhere on the page.

We do a binary transfer of the print file to our HP3000 using a file size of 256. We end the HP3000 filenames with the letters PCL to remind us that they

are in HP's printer command language. For logos with more than one size we add a number to indicate the size. For example, DLPCL1 is the Dairylea one inch logo. After the transfer, we run Fantasia's RASTCOPY program to automatically do the PCL editing. Besides getting rid of the commands you definitely need to eliminate, RASTCOPY also cleans up the file by eliminating unnecessary commands, producing more efficient code. Next we run Fantasia's FORMCONV program to turn the output from RASTCOPY into a form.

We easily accomplish the third step, printing the logo or signature where we want it, with Fantasia by including a GOTO command followed by a FORM command referring to the file created by FORMCONV. Fantasia tracks whether the logo or signature has already been downloaded to the printer and automatically downloads when necessary.

If you do your own PCL coding you need to clean up the print file created by your word processor before you transfer it. Use the appendix of your LaserJet manual and look at the beginning and end of the file to determine which commands to delete. Generally you need to delete commands up to the first <Esc>* command and after the last <Esc>* command. To print, you must send the file to the printer with carriage control of *no space control* (208). To do this with COBOL, include a special-names section to define no space control (example, NO SPACE CONTROL IS NO-SPACE) and use the definition in your write statement (example, WRITE OUTPUT-RECORD AFTER ADVANCING NO-SPACE).

With PCL code, to place your logo or signature where you want it, write cursor positioning commands in the line before your graphic PCL begins. Your programs can read your transferred graphic PCL files and write them each time you want to print the graphic. Alternatively you can read the lines into a table when your program begins and print them from there each time. Either way, the entire graphic goes to the printer for each page. Depending on the size, this can slow your printer down considerably. Instead, you can add PCL macro commands to the file you transferred from your PC and write a simple program to print the file to the printer to set up the graphic as a macro. Then your programs only need to include one simple PCL command to call the macro and print the graphic. You must make sure you have sent your macro/graphic file to the printer before you want to use it, and don't reuse macro id numbers for different graphics or you may end up with a great deal of confusion.

Another option for including logos and signatures is to turn them into fonts. You can get PC software to use with a scanner to create the special font. While I have not tried this approach, I have heard it has an advantage because you can edit the scanned image in the PC software and the end result may take less space than a PCL raster image. You can contact Proactive Systems to find out more about this approach. If you do not have a scanner, you can have an outside

party create the logo and signature files for you, but then you lose the flexibility of making your own changes.


Potential Pitfall

Don't let your signature or anything else go into the MICR area.

No printing other than the MICR encoding can appear within the bottom 5/8 inch of the check. To be safe, we stay a little farther away than the 5/8 inch. I once heard of a company that let their signature run into the MICR area. They had to pay their bank a penalty for each check that came through that way, and since they had printed thousands, the cost was significant. To avoid this problem, I include the signatures when I send test checks to the bank. We print shading behind the signatures to make it look more like a signature plate.

Check Size

The size of a check has to fall between a maximum of 8 ¾" x 3 2/3" and a minimum of 6" x 2¾". Most people's personal checks are the minimum size. If you are going to print checks on a LaserJet, the natural width for the check is 8½". You could print checks on legal paper in landscape mode and get two 7" checks side by side. If you are considering the legal paper approach, you might want to check paper prices first. Legal paper might cost more, because it is not commonly used for check stock. If you only want to print checks with no stubs, letter size paper divides evenly into three 3 2/3" checks or four 2¾" checks.

Two factors were important to us at Dairylea for determining check size. First, we wanted the check stub to be as large as possible. Second, we wanted the folding machine to fold them on the perforation and have the name and address show through a standard window envelope. We tested the folding machine with plain paper and decided on a 3¼" check. The number and placement of perforations had no impact on the check safety paper pricing.


Potential Pitfall

Make sure you fold on the perforation.

Perforations on check safety paper designed for laser printers are not as well defined as they are on paper designed for line printers. The normal perforation ridge for line printer checks would create jams in a laser printer. Since the perforations are not as well defined, you must fold the page on the perforation before you can tear the check from the stub. If it's not folded on the perforation, the check or stub can tear. You'll save the people receiving your checks grief by folding them correctly.

Potential Pitfall

Beware of reversals in packages of check safety paper.

We have encountered packages where half of the package has the top at one end and half at the other. With a standard layout of a check at one end, if you simply dropped the whole package in the paper tray, half your checks would print on the stub end. This would make it very difficult to tear the check from the stub.

Our solution is to have the check safety paper perforated at both ends. Since the perforations are not very noticeable, no harm comes from having a perforation in the stub, and we eliminate the need to make sure that the top is always in the same direction.

Another benefit of perforating at both ends is that if you desire, you can print some checks duplex and others simplex in the same run. With duplexing, the back side of the check prints first due to the paper path of the printer. That makes the normal top of the sheet the bottom, so with a perforation at only one end, the duplexed check would be on the wrong end of the paper. Perforating at both ends gives you more flexibility.

Fixed and Variable Formats

One nice thing about not being restricted by preprinted forms is that you can produce checks stubs with variable formats. For example, look at the check stub in illustration 2 on page 7. Tables near the bottom show deductions from the check and milk pickups. The number of deductions and pickups vary from check to check. We can print the size of the tables to match the number of lines for each check. If we had to leave space for preprinted tables for the maximum number of deductions and pickups on every check, we would quickly run out of space.

At other times fixed stub formats do make sense. For example, illustration 4 on page 14 shows a stub from an accounts payable check. Not much varies on an accounts payable check, except the number of detail lines. By keeping the table a constant size, the stub looks balanced, even if only one detail line appears, and the empty space in the table doesn't detract from the overall appearance.

When designing checks for line printers, you must decide where you want all the lines and boxes and have them preprinted. While you can print straight horizontal lines on a line printer, you can't do much more. With LaserJets, you can print lines, boxes, shading, etc., with PCL as you go, wherever you want them. If you haven't done much PCL coding, be aware that it can be quite complicated. Since we use Fantasia at Dairylea, we do not need to get down to

Dairylea Cooperative Inc.

DAIRYLEA COOPERATIVE INC.
P.O. BOX 4844
SYRACUSE, NEW YORK 13221-4844

Vendor No. 06-173275 Check No. 6A-104480

Invoice Date	Invoice No.	Batch Voucher No.	Gross Amount	Discount Amt.	Net Amount
05/01/93	97619	90005 00497	81.08	.00	81.08
05/01/93	97620	90005 00498	81.08	.00	81.08
05/01/93	97618	90005 00499	81.08	.00	81.08
			243.24	.00	243.24

Illustration 4: Accounts payable check stub sample

the nitty gritty of PCL. We use simple Fantasia commands like LINE and BOX, which Fantasia translates into PCL for us. Yes, we could do the PCL coding ourselves, but Fantasia saves us a great deal of time and provides functionality beyond what we would be able to do ourselves. For example, I would not want to do the coding to print our deduction and pickup tables that use proportional fonts. On my own, I could produce all the basics I need for checks, but I could not produce some of the fine enhancements.

Making the Change

Now that we understand how to create the key ingredients, let's consider how to put them all together to convert line printer checks to LaserJet checks.

Conversion in Stages

Several conversion options are available when making the move to LaserJet checks. You can take a big leap to free format checks, which requires designing a basic check form and modifying your source code to add the variable check information. Or, you can design a stable layout similar to your existing checks but take advantage of some of the basic LaserJet features such as different fonts for different parts of the checks. This also requires modifying your source code, but the changes are simpler. Finally, you can design a check identical to

your old check, select a font character spacing to match, and you may not need to change your source code at all. You simply print your lines onto the new form.

At Dairylea the approach we took depended on the check type. We never tried to match old checks exactly and just let the old source code print on the new form. We did begin with a matching form for one of our invoices, though we eventually rewrote it to take advantage of more advanced LaserJet capabilities.

One nice thing about LaserJet checks is that it is completely practical to make the changes in a series of steps. You can start by making minor modifications such as varying fonts and including the MICR information, but leaving your stub layout the same. Later you can enhance your stub information by adding or varying tables, or doing whatever makes sense for your company.

What to Do When You Can't Change the Source Code

At Dairylea we produce our accounts payable checks with an old, horribly written purchased package. While we have the source code and theoretically could make changes to it, we avoid doing so when at all possible. We wanted LaserJet checks, but we didn't want to change the source code.

We left the program alone, but changed the file equation in the job stream to write the check file to disc rather than to the printer. Then we added a program to the job stream that reads the disc file and adds the commands to create LaserJet checks. Besides creating nice LaserJet checks, another benefit relates to multi-company checks. The accounts payable program creates checks for multiple companies, opening a new printer file for each company so it prints on the proper checks. To the file equation, we added ACC=APPEND, which puts all the companies in the same disc file. We only need to process one file, and all the company information changes automatically. The blank check stock is all the same, so there is no need to create multiple spoolfiles.

We developed the program to read the accounts payable check file and create LaserJet checks by analyzing a file created by the accounts payable job. We saw what information printed where and on what line, and then wrote the program logic based on that.

Another benefit of this change is that the accounts payable program produced non-standard sized checks, making them expensive, with the stub before the check. Since our people prefer the check first for the folding machine, the program we wrote holds the stub information until it encounters the check information, then prints the check followed by the stub. Also, because the old checks were short, when the detail information does not all fit, it voids the first check and continues on the second. When our program encounters the void information, it simply throws it into the bit bucket and combines the detail all on one check stub.

Potential Pitfall

Who knows what that program really does?

While we examined a whole series of actual checks and talked to accounts payable personnel, when we actually started using our program, we encountered a few unexpected circumstances. One was when a check appeared with a name but no address. Since we didn't expect such an event, our program didn't handle it properly. It was a simple matter to modify our program and rerun it on the same accounts payable disc file. We simply threw out the unacceptable batch of checks, printed the corrected ones, and the actual accounts payable system was unaffected. There was no need to restore files and rerun checks.

So, when you create a process like this and don't really know all the ins and outs of the real program, keep a close watch on the checks for the first couple weeks or months, and be prepared to make changes to your program.

Printer Speed

Factors you need to consider concerning speed are the total number of checks you print, how many you print at a time, and how critical the timing is. Do you run your checks all the same day, or do you spread them evenly throughout the month? When you run checks, do you need them immediately, or can you produce them ahead of time? It really doesn't matter if it takes all day to print the checks if you don't need them until tomorrow.

I had been wanting to make the change to LaserJet checks for some time before we actually did so. One of the things holding me back was the eight page-per-minute printing speed of the LaserJet printers. While we print some of our checks in smaller quantities throughout the month, our most critical time comes when printing members' final checks. There is a very small window between the time when the member payments department receives all the information they need to produce the checks and when the checks need to be in the mail.

The LaserJet IIIsi, printing seventeen pages per minute, was the solution for us. While it takes slightly longer to print one check with the LaserJet than it did with our 900 line per minute printer, the total time has decreased significantly, because we have eliminated the setup time and problems, and because we no longer need to burst and sign the checks.

Another option, which I had not considered at the time, was to print checks on more than one printer. To do this, write your program to close the print file and open it again after every so many checks. Then, you can print each

spoolfile on a separate printer. With multiple LaserJets you can print more checks per minute than you can with a single line printer.

Another consideration is the printing complexity of the check. The pages per minute rating of a printer means that is the *fastest* a printer will go. It doesn't mean it will always print that fast. If your page is too complex, the printer will slow down. For example, we print pages of bar code labels on LaserJets. Whether we print them on a LaserJet II or LaserJet IIIsi, we only get two to three pages per minute. The computer simply cannot send the commands over the serial connection any faster. With our checks, we do achieve the maximum print speed.

To keep up the speed of the printer, minimize the amount of data sent with each check by keeping the standard information in the LaserJet. Fonts can be resident in the printer, in an installed cartridge or previously downloaded as permanent fonts. Download logos, signatures and the non-variable part of the check form ahead of time as PCL macros.

At Dairylea we have a reset file that contains all the soft fonts, graphics and forms we use. Every time we turn the printer off, the first thing we send to it when we turn it back on is this file, so everything we need is always available. Fantasia keeps track of all the macros and fonts for us, and our reset file is really just a list of fonts and the forms we created with Fantasia. You could do the same thing with your own PCL commands. It would just be a lot more work.

PC Potential

A number of companies sell special hardware and/or software for personal computer solutions for printing checks on laser printers. While you probably could transfer information from your main computer to a PC to produce checks, it makes more sense to produce your checks on the machine where your information normally resides. At Dairylea this normally means the HP3000, but we do have one case where the information is coming from a PC, so we produce these checks on the PC with some help from the HP3000.

Dairylea has an insurance agency subsidiary called Agri-Service Agencies Inc. They send refund checks to subscribers when they make certain insurance changes. For a number of reasons, they did not want us to write a program to compute the refunds and produce the checks on the HP3000. They use a PC worksheet to compute the amount of the check. Then someone typed the checks, made copies of them for the file, and finally another person entered them as manual checks into the accounts payable system.

To eliminate the manual work and print the checks on LaserJets, we added a macro to their worksheet to accumulate the information needed to produce

checks and an accounts payable batch. When Agri-Service personnel complete a batch, accounts payable retrieves the file from a network drive and runs macros to add check numbers, dates, etc. and create files for the accounts payable system and check printing. Next, accounts payable uses word processor macros to clean up the files produced by the worksheet program, to print checks and to print paper copies marked *COPY-VOID*. The check template on the word processor includes PCL commands that call the forms produced on the HP3000 with Fantasia. It does take a number of steps on the PC to produce checks and an accounts payable batch this way, but the steps are a lot less work than the manual routine and eliminate some possibilities for typing errors that can occur with a manual system.

How and where PCs fit into your check writing systems depends on a number of factors, but this example shows one way PCs and the HP3000 can work together to provide a solution for a unique situation.

Check Numbering

With preprinted checks the numbers are preprinted for you. Each time you print a set of checks you have to verify the starting check number. It you print several kinds of checks on the same bank account you can set aside a series of numbers for each type of check, but when you run out of checks you may have problems with check number breaks. Alternatively, you can order checks with different series of numbers for each type of check. The disadvantage of this is the additional checks you need to stock.

With laser printed checks you don't have problems with matching up your check runs to preprinted numbers. In fact, I believe that you could start every check run with the same number. If you do, your bank reconciliation will be a big challenge, and you'll have quite a task if you need to find a certain check, but the bank won't care. They should all clear just fine.

What we did with check numbers is to assign series of check numbers to the different types of checks. For example, accounts payable check numbers start with a 1. Final milk checks start with a 7, etc. This way each group is responsible for tracking and entering their own starting check numbers, and there is no danger of conflict.

The number of digits in a check number varies. Check with your bank to determine the correct number of digits, or simply look at your existing preprinted checks and use the same number of digits.

Security

The thought of printing complete checks including signatures from blank paper on LaserJets scares some people from a security standpoint. The question,

"How can we keep people from just printing checks for themselves when it's so simple?," quickly comes to mind.

While this can be a concern, the real security issues do not revolve around your printing technology. They relate to your accounting and physical controls. The questions to ask are: What are our current check controls? What will the true impact of the printing change be? Are current procedures any more secure than the proposed changes?

For example, who runs your check programs now? Even though you may make some program changes, the basic procedure will remain the same. If the opportunity to run checks is controlled now, it will be with the new checks. Only authorized people should be able to run checks.

If you sign your current checks with a check signer, somebody has access to the signature plates. Is it the same people that run the checks? If so, then what is the difference if they simply preprint the signature and eliminate a manual step?

You can print your checks on LaserJets without the signature if this is really a concern, but you will lose the benefit of eliminating the extra step of check signing. Similarly, you can download your signatures as temporary rather than permanent. Some check writing laser systems offer passwords on the signatures or some type of physical control. On the surface, these seem to be good security features, but consider this. Does anybody print the same signatures on letters or other documents? If they do, what will prevent them from printing the signatures on checks?

Don't make this too easy for people. For example, I had written a word processing macro for the accounts payable person to print manual checks rather than typing them. She feeds the preprinted check manually into the LaserJet, and it prints all the same information she would have typed. She was excited about this change, because it was easier than typing the checks. A few days later she came to me and said her boss, the controller, had asked her to check with me about printing the signature also. I told her to send him over to talk to me about it. I said to him, "Think about it. This person is printing these checks from a word processor. There are no controls. She can type a check with any name and address for any amount. Do you really want me to print a signature?" "I guess not," he replied.

If people want to be clever, they can use the computer and LaserJets to forge checks, but they can also forge checks manually. Forgery is against the law, and the way you print checks isn't going to make people more or less honest. However, you don't want to make creating checks too easy and, therefore, more tempting to people.

Another question to consider is, who currently has access to your preprinted check stock? In many ways it would be much easier for somebody to take a preprinted check, write it out to themselves and sign it than it would be for them to create a complete check from a blank piece of paper.

These are the kinds of issues you need to examine. Overall, printing checks on LaserJets is not more of a security hazard than using preprinted checks. The concerns primarily come from a normal fear of change and from not having thought through all the implications.

At Dairylea only authorized people can run check programs. We keep the MICR toner and check safety paper locked in the computer room with the LaserJet we use to print checks. Before we had a dedicated check printer, we would bring the toner and paper out of the computer room to the printer and return it when the checks finished printing. After we print the checks, we give them directly to the authorized person.

When I present this paper at the conference I will talk about some security concerns that I am reluctant to put in writing. Again, these issues exist no matter how you print your checks.

Summary

In summary, what do you need to do to print your checks on LaserJets?

- Decide on overall creation strategy
  Will you write your own PCL or use a software package?
- Contact your bank for check requirements.
- Make check paper decisions
  Choose a paper supplier, perforation location, color, etc.
- Meet MICR requirements
  Choose MICR toner supplier and MICR font.
- Change your programs and/or procedures
  Using the tools you've chosen, design and create your new checks.

If you spend some time exploring your options, you'll find that making the change to laser printed checks will be one of the better moves you can make.

References

MICR toner from MICRTech P.O. Box 152 102 S. Main Street Brownstown, Indiana 47220 812-358-5400
Fantasia software from Proactive Systems 1411 South Woodward Bloomfield Hills, Michigan 48302 313-333-7200

Paper 7005

A Network to Support Change

Robert P. Hillseth
Epson America, Inc.
20770 Madrona Avenue
Torrance, California 90503
310.782.4154

CUSTOMER SUPPORT CENTER

In late 1991, Epson America restated its vision statement to make Customer Satisfaction the number-one priority. Emphasis was shifted to include end-users as primary customers in addition to resellers. The company reinstated its 800 support number, added services such as literature fulfillment by phone, and implemented on-site warranty service for its PCs. But true improvements to customer satisfaction require more than increasing telephone lines and adding voice-prompted options. It requires a level of support that makes people glad they own or operate an Epson product.

Seeking dramatic change, Epson examined the methods used to provide customers with products and services. They learned that it was difficult for end-users to get information prior to buying as well as post-purchase assistance. Several different computer systems, telephone systems and outside service providers were used with varying degrees of success. Callers typically had to wait minutes before speaking with a representative, and sometimes got referred to another number.

Epson has re-engineered its customer support facilities by establishing a Customer Support Center (CSC). Its mission is to increase customer satisfaction, brand loyalty and encourage the purchase of Epson products and services. This is being accomplished by courteously and effectively providing customers with accurate dealer/service center referrals, product information, technical support and a source to purchase accessories, computer systems and customer replaceable parts. Its vision was a CSC from which users could get all the desired information through a single telephone call. This meant enabling CSC representatives to pickup calls quickly and answer the question, complete the transaction or solve a problem while the customer remains on the phone. In addition to training, the representatives must have quick, easy access to thousands of documents about the company's products, dealer locations and service centers. By creating a matrix of the products and services offered and the method used to supply them, Epson had a clear picture of the systems already in place, those that needed to be created and those provided by outside vendors that would be brought in-house.

The Information Systems (IS) group was tasked with providing the computing systems required to make the Customer Support Center a reality. The computing environment at Epson's Torrance, California, headquarters includes an IBM 3090 mainframe for finance, accounting and distribution systems, and Hewlett-Packard 3000 machines for electronic mail

and accessory and direct marketing systems. Access to these systems is provided to Epson desktop workstations via a LAN. IS had to find a computing platform that would provide an interface to the various computing resources in use or expected to be used, including networked PC software packages and the IBM and HP systems. Specific goals were established for the interface: It had to be easy to use, easy to support and maintain, provide rapid switching between applications, be available 24 hours-per-day/seven days-per-week, and allow new applications to be added easily.

INITIAL STARTUP

Implementation of the Customer Support Center was accomplished in phases over a number of months. The initial phase had the center staffed to provide pre-sales support, which includes dealer referral, literature fulfillment and answering questions about products being considered for purchase. Later phases, currently provided by separate systems or outside vendors, will incorporate accessory product ordering and technical support for end-users, resellers and authorized service centers.

Dealer referral and literature fulfillment systems were developed in-house on the IBM 3090. The referral system allows CSC representatives to provide names of dealers in the caller's area who carry the desired product, as well as the nearest service center. Requests for literature are fulfilled daily through a mailing package on the system.

The second facility provides pre-sale technical support, by which customers can get information involving product specifications, competitive comparisons and pricing. Prior to setting up the Customer Support Center, this information was available in printed form. Storing it into a PC-based image-management package allows it to be accessed quickly and insures the most current information.

A fax server has the capability to transmit anything that can be printed. When a customer requests information by fax, it is sent almost immediately by the system.

PLATFORM

The IS group developed a LAN as an interface to the various resources in use or expected to be used, including the IBM and HP systems and PC software. The group identified several goals for the

selection of the interface. It must:

1. Be easy to use.
2. Provide rapid switching between applications.
3. Have the ability to add applications.
4. Be easy to install and maintain.
5. Be available 24 hours a day, seven days a week.

Microsoft Windows was chosen as the integration tool for managing the presentation, switching among the applications and allowing terminal emulation access to the HP and IBM systems. The LAN would consist of 20-plus workstations, a file server supporting networked PC applications, supplemental print servers, a fax facility, CD-ROM library access, a backup facility, routers and gateways to the IBM and HP systems. The initial startup of the center called for 15 hours-per-day and five days-per-week operation, going to continuous access as soon as support personnel could be trained.

STARTUP APPLICATIONS

The dealer referral/literature fulfillment system is a CICS application. Rumba software from Wall Data was chosen as the 3270 terminal emulator running under Windows. Rumba's "Profile" option allows multiple copies on each workstation that are runtime versions with a fixed configuration. This avoids user-modifications and maintains a known workable configuration. Keyboard remapping was used to match some of the functions DOS users had and, at the same time, retain the traditional keys used by Windows.

Reference materials for the Customer Support Center representatives comprise thousands of pages of technical and support documents, reference manuals and user manuals. A library-management system accesses the text and graphic images quickly and in a variety of ways.

In addition, about 100,000 pages of pre-sale and post-sale information must be readily available in response to customer questions. Fifteen years worth of technical specifications, comparative model information, user manuals, repair manuals, parts lists and product reviews need to be in the database. After reviewing several PC-based packages, Epson chose File Magic by Westbrook Technologies. File Magic was particularly good in creating multiple document retrievals that can select the same document.

With the initial applications selected, IS began looking at hardware and network components.

HARDWARE SELECTION

Standard Workstations:

Epson's IS group chose to standardize on a single workstation that can support applications planned for the initial startup as well as in the future. Epson has a number of PCs that are good platforms for Windows, testing revealed which would be best for Customer Support Center applications. All the network software and applications were loaded and tested on each of the PC candidates using a test Token Ring network.

The image-management application was the most resource-intensive, and Epson's Equity 486SX/25 PLUS fit the bill. Each system was equipped with 8 Mbytes of memory, a 3.5-inch floppy drive, a 100 Mbyte hard drive, a 17" monitor and a mouse.

To avoid support headaches that arise when users have dozens of different PC models and several versions of MS-DOS, Epson standardized the entire network on the Equity 486SX/25 PLUS. Each user was given the same hardware configuration, with common software residing on a file server. The determined configuration -- including the directory structure, DOS config.sys and autoexec.bat files -- was placed on a separate directory.

A cloning process was developed to standardize and populate the workstations for each user. This process involved booting the PC from a floppy that also contained the network software. The installer logged onto the network as a user named CLONE_WS. Then the system issued a warning that it was about to erase the files on the workstation and replace them with the clone version. After receiving confirmation to proceed, the cloning process would erase the files and directory on the workstation, create new directories, and copy the appropriate files (including MS-DOS) from the file server onto the workstation.

Now each workstation was identical with a known workable configuration. Cloning takes about three minutes, including two minor processes to make addresses to the IBM and HP systems unique. The installer could then reboot the system and test connectivity to IBM, HP and LAN applications. Where it may have taken hours to install and configure

all the required software, the process was reduced to just minutes. This supports the IS goals of being easy to install and maintain. The same process will be used on an existing workstation that becomes inoperable due to lost or corrupted files. An added benefit of cloning is the hours saved in troubleshooting. Each configuration is identical, known to work and, if necessary, can be recreated in a few minutes.

A second level of backup is possible if a workstation fails to operate properly after being recloned. Because each user has an identical hardware configuration, a spare PC can be configured, cloned and connected to the network. The workstation PC, monitor, keyboard and/or mouse can be replaced within several minutes. In a worst-case situation, a Customer Support Center workstation should be back in operation within 20 minutes. This simple backup process is expected to greatly contribute to user uptime and reduced support costs.

File Server:

Epson's 486/33 EISA Series tower PC was selected as the file server. This system has the memory and expansion capacity to store volumes of data. The Customer Support Center system is configured with 64 Mbytes of memory, a 3.5-inch floppy, a monitor and EISA SCSI interface and network interface cards.

About 100,000 pages of text and graphics had to be stored for online access. This image application had by far the largest data storage requirements. A sample of the different documents was stored in compressed format and from this it was estimated that approximately 7 Gbytes of online storage would be needed.

As system availability and speed of response were high on the priority list, mirroring, duplexing and disk arrays were evaluated as potential storage media. Ultimately, the IS team chose to connect five Micropolis Raidion RM1750 models, each with 1.75 Gbytes of storage, for a total of 7 Gbytes of usable space. The system provides RAID Level 5 fault tolerance, including the ability to swap a failed drive and rebuild lost data in the background without having to shut the server down. As a result, users can be productive while the failed drive is being replaced. A sixth drive was purchased as a spare to swap in the event of failure. With Micropolis' flexible backplane installed, the backup disk was plugged in so the software can switch automatically to the hot-backup unit and rebuild lost data. This can be accomplished automatically without operator intervention. Along with fast access to the library of graphic and text images, the disk array can also

be expanded to 96 Gbytes.

To increase system integrity, a second, identical file server was installed for use only if the primary unit fails. The Raidion disk array is connected to the EISA SCSI controllers via a switch box and can be easily connected to the backup file server if required. While failure is not expected, the critical functions performed require a backup that can be operational in minutes.

Scanning Workstations:

About 100,000 pages of documents were scanned and indexed into the File Magic database. Due to the volume of data, Epson used Fujitsu's M3096E scanner with an automatic document feeder and the ability to scan 11x17-inch documents. It is driven by a Kofax 9250 interface. The source documents were scanned and stored locally before they were indexed and stored on the file server. Each system has 16 Mbytes of memory, a 3.5-inch floppy drive, a 340 Mbyte SCSI hard drive, an 3.5-inch optical SCSI floppy drive and a 17" monitor.

The 340 Mbyte SCSI disk stores scanned documents for quality-control testing before they are indexed and stored on the disk array. The optical drive holds 128 Mbytes and is used as a source document backup device. Typically, all scanned documents that relate to a single Epson product will be stored on this removable device. The hard drive and optical drive are attached to the same EISA SCSI controller card.

The scanning workstation uses the same 17" monitor configured for 1028x768 resolution. This allows the database administrator to perform quality assurance on each image just as the Customer Support Center representative will view it.

Hardware Standards:

Along with standardizing workstation applications on the Equity 486SX/25 PLUS, a 486/33 Epson EISA Series machine was chosen for applications requiring exceptional throughput. Desktop and tower models are used as file servers, scanning workstations and TCP/IP routers.

Although the fax and print servers ran successfully on an Equity 286 PLUS PC, the 486SX model was used to maintain the standard. While this provided more computing power than necessary for some of the servers, it fit with the IS department objectives for network availability and reduced

support costs. Even though a 486-based unit costs significantly more than a 286, money was saved in the end by eliminating the need for a 286 backup PC.

Fax Server:

The ability to send and receive faxes from a workstation is an important tool for the Customer Support Center. Transfax's fax management software application resides on a dedicated PC along with multiple GammaLink fax cards. The Transfax package is in reality a print redirector. Any software application can be used to create the print image and send it to the "Transfax Printer" as selected from the Windows control panel. A pop-up window prompts the user to enter a name and phone number or select from a fax phone directory. Once the information is entered, the image is uploaded to the server.

The fax server adjusts the print density to the appropriate fax density, attaches a cover sheet, dials the number and completes the transmission. The server is configured to retry multiple times in the event of a busy signal, paper outage, or paper jam. It will also send the same document to a group of addressees if requested. To verify response to the caller, a Customer Support Center rep can check the transmission status through a screen inquiry. Although this feature is not currently used, the fax server can schedule deliveries at a specific time in order to take advantage of off-hour telephone rates. The initial configuration has two fax lines, both configured to send or receive.

Since the initial implementation the system has been expanded to three fax processing systems with a total of twelve fax cards to accommodate the volume of faxes. One of the features we are planning to test in the near future is the DID facility of the GammaLink boards and Transfax software. This will provide each use or group of users with their own fax number and the system will deliver inbound faxes directly to the end user's pc without any intervention. The user will be notified of a received fax with a network message.

Print Server:

A dedicated print server was installed using Netware's dedicated print server program. An Epson EPL-8000 and EPL-7500 laser printers were attached to an Equity 486SX/25+, providing both PCL5 and Postscript printing facilities for users. The 486 machine was chosen as part of the standardization plan.

NETWORK SOFTWARE

Connectivity to the IBM Applications:

Connectivity to the 3090 mainframe is provided using Novell's Netware for SAA that runs as an NLM. Being in a Token Ring environment, the SAA software connects the user to a 3174 communications controller. SAA can operate on the file server along with other Netware services. For maximum performance, a separate dedicated communications server using a runtime version of Netware 3.11 was installed. To provide redundancy, two SAA servers were setup to split the load split between the CSC representatives.

To accommodate multiple shifts among Customer Support Center Employees, the two servers have been allocated accordingly. In a worst-case situation, half the devices would be non-functional during which time an effected user could occupy the workstation of an off-shift user. Failures are expected to be short in duration, as a backup PC is already configured and available.

Connectivity to HP3000 Applications:

Walker, Richer and Quinn's (WRQ) Network Series software provides the connectivity to the two Hewlett-Packard 3000 machines. WRQ's 3000 Connection software, along with their HP terminal emulation, Reflections for Windows, serves as a package for connecting multiple sessions to multiple hosts. Each PC on the Token Ring network connects to the HP3000 as a virtual terminal session. Novell's Multi-protocol router software provides the routing capability between the token ring and the ethernet networks. The 3000 Connection software provides the NS/VT support and talks to HP's NS Services.

Several applications run on two 900 Series HP3000s, and this software provides simultaneous access to both systems as well as multiple sessions on a single host if required.

Epson's CSC LAN connects 200 PCs from a Token Ring environment running under Novell's Netware 386/3.11 operating system. The HP3000 has an 802.3 Ethernet LAN. Novell's Netware MultiProtocol Router connects the Token Ring and Ethernet networks. The router is based on NetWare Runtime 3.11 and is installed on a 486/33 Epson EISA Series desktop. The system is configured with 16 Mbytes of memory, a

3.5-inch floppy drive and a 100 Mbyte hard drive. A second identical server was installed as a backup.

Backup Server:

Epson's IS team needed the ability to backup its 7 Gbyte file server in an unattended manner and in a relatively short period of time. After evaluating several tape backup systems with auto changers and optical jukeboxes, Micro Design's LaserBank Jukebox system was selected. It includes a Hewlett-Packard 10 Gbyte optical jukebox, Netware software drivers and backup software. An 486/33 Epson EISA Series tower is used as the server and is configured identically to the file servers except for the number of SCSI adapter cards. The jukebox has a single drive and slots for 16 Epson 5.25-inch rewriteable optical disks. It is multifunctional and can accept both rewriteable and WORM cartridges.

SCSI Express is the software package that runs as a NLM on Netware and provides the interface between Netware and the jukebox. Their LanLibrarian package is used as the backup software.

SUBSEQUENT PHASES

Additional Users:

During 1992, other departments were added to the network. What initially started with 20 users has expanded to several hundred as the end-user and reseller support groups were trained on the new applications and added to the network. To handle the load the file servers were upgraded from 486/33 EISA systems to 486/50 EISA systems and a second and third file servers were added. A second Micropolis 7Gbyte disk array was also added.

In early 1993, Epson launched Epson Direct to sell systems directly to end users. This new group was added to the network and one of the two 950 was upgraded to a 960 to handle the additional application load on the HP3000 systems.

Call Tracking:

In June 1992, the call-tracking application became operational and serves as a central facility for recording customer requests. Epson will use it to measure problems, track its ability to correct them, follow-through on

problems that can't be resolved by a support rep, and evaluate the overall effectiveness of the Customer Support Center.

CD-ROM Services:

Eight CDROM players were added to the Jukebox file server. The SCSI Express software that run the optical jukebox also supports SCSI CDROM devices. A number of CDROM titles were made available to our customer support representatives. These included Computer Library, Support on Site for Applications, Novell Support Encyclopedia, Select Demos, Hewlett Packard's MPE documentation and several Microsoft CDs.

NETWORK EFFECTIVENESS

One of the goals was to reduce the various telephone numbers customers have for contacting Epson. The first application, Dealer Referral/Literature Fulfillment and Pre-Sales Technical Support provided the customer with a single 800 number to answer any question about buying an Epson product. Previously these functions were handled by an outside vendor and Epson resellers. As post-sales support for end users and resellers is incorporated into the CSC several more 800 numbers will be consolidate into a single number. Each CSC rep will have all the tools available to them and will be able to answer any question. This has dramatically reduced the number of calls being transferred which is a source of customer dissatisfaction.

The network provides the facility to easily switch between a number of software applications while the customer is on the phone. This has increased the total number of CSC reps who can handle multiple problems where we used to have specialized functions who only dealt with their area. As each rep has been trained on all functions and has access to each, Epson is now able to provide common business hours for all customer service functions. The ability to switch between applications and the speed of the response has enabled the CSC to answer 98% of the calls that is a substantial increase when the service was located at an outside vendor. In addition the average wait time for a customer in the telephone queue has been reduced from several minutes to under 10 seconds. The average length of a call has been reduced from five to six minutes to just over three minutes as the rep can quickly get to the required information.

By implementing our dealer referral system in-house we were able to tie into the distribution system and located dealers who have recently

purchased the particular Epson product that the customer is calling about and are located near the customer. This provides accurate current information and a great improvement over the outside vendor's system.

The image-document management system has provided electronic access to thousands of pages of technical documents, product brochures, price lists and manuals that can display all the information required in a matter of seconds. What used to occupy rows of book cases can now be retrieved electronically. More information can be accessed than was possible with paper manuals. An additional benefit is the consistency of the information. There is now a single source of the data and it is always up to date. Hundreds of man hours have been saved by not having to update multiple copies of the documents. The customer receives accurate information that is consistent regardless of whom he speaks with. Floor space requirements have been greatly reduced by the elimination of all the paper.

The fax server provides the ability to fax any information, data or graphic picture that can be printed without leaving the application. The reps fax hundreds of documents a day at a savings of about 10 minutes per fax. The customer usually receives the fax while on the phone with the CSC.

The best measure of the effectiveness of the network can be summed up by a quotation from a letter addressed to the president of our company from a NEW customer who called the CSC. "Not in the near past have I encountered a more pleasurable person to do business with. He was courteous, knowledgeable... I have since purchased an LQ-570 Epson printer.".

INDUSTRY RECOGNITION

In October 1992 at NetWorld in Dallas, Epson's Customer Support Network won an ENNE Award. The Enterprise Network Excellence Award (ENNE) was created by NetWorld and Network Wold to recognize achievements in networking that improve corporate efficiency and productivity and to reward those who understand that information management is a strategic corporate asset. Epson's nomination was sponsored by Novell.

In April 1993 at the Association for Information and Image Management (AIIM) show in Chicago Epson was awarded the Silver Award for excellence in Imaging. BIS Strategic Decisions sponsors the

Imaging Excellence Award. This award is designed to recognize a user organization that has successfully applied imaging technology to meet strategic business and organization objectives.

In the summer of 1993 Epson was a finalist in the Computerworld Smithsonian Awards program in the Business and Related Services category. This program recognizes exceptional and innovative use of information technology and a level of commitment to advancing the use of technology.

#7006: Data Archival with Optical Technology

Husni Sayed
Deborah Littlefield
Joe Nemeth
IEM, Inc.
1629 Blue Spruce Drive
Fort Collins, CO 80524
(303) 221-3005

1 Introduction

As an installation grows, it fights a constant battle between the need to maintain all of the information it has accrued, and the need to reduce the costs of storing a constantly increasing amount of data. As time goes on, more and more space is devoted to storing information that is less and less frequently accessed. While the information is not needed on a constant basis, when it *is* needed, it is needed now! Most installations cannot afford to keep such data using premium system hard disk space, yet keeping it on tape is inconvenient and time-consuming to retrieve.

The development of optical disk technology is revolutionizing some of the traditional categories of computer data storage. The appropriate use of optical technology is *not* as a replacement for either hard disk or tape technologies, but in a niche which requires both the unlimited capacity and the random access capabilities of these drives. These capabilities are a perfect fit for storing archival data. With this technology, users can have access to archival information within seconds or minutes, instead of hours or days.

2 The Information Explosion

At one time or another, every computer installation that has been around for more than a few years begins to experience an information explosion. Early on, an installation encounters few problems finding places to store all of its information. The explosion occurs as an installation grows—more and more work is done, more and more projects are completed—producing massive amounts of information that must be maintained. Typically, there are three classes of information which must be maintained: current information, backup information, and archival information. These three classes of information are discussed below.

2.1 Current Information

Current information is information that is both stored and accessed on a regular basis, to which users must have virtually immediate access.

- **Current information is constantly stored and updated as it is changed by the user.**
- **Current information is continually accessed by users, and must be instantly available upon request.**
- **Current information is always dynamically changing as users create, modify, and remove files.**

Current information includes such things as system files needed for daily operation (the operating system, editors, etc.); databases; records on current employees, students, or patients; and files that are needed for current projects. The amount of current information on a system stays relatively proportional to the number of users: as new files are added to the store of current information, other files become obsolete. Therefore, a storage device for current data does not need to have unlimited capacity.

> **A device used for storing current information must give users the ability to quickly and easily create, edit, modify, and remove information. This must be a fast, random-access device.**

2.2 Backup Information

A backup is a "spare" copy of all of the information stored on the system, in its most up-to-date form (as of the time the backup was made).

- Backup information is stored on a regularly scheduled basis, so that the system can be restored to its normal operating condition in the case of a disaster (major or minor) such as a disk crash, or inadvertent deletion of a critical file.
- Backup information is retrieved rarely, most often when a system or device goes down. The smallest unit that might be retrieved is a single file.
- Backup information needs to be available, but does not need to be instantly accessible.

The size of a backup will depend upon the backup method used by an installation. If an installation performs only full backups (copying all of the information on the system), then the backup information is equal to the size of the system. Some installations perform a full backup at regular intervals, and incremental backups (storing only what has changed since the last backup) in between. In some cases, incremental backups are impractical—if a database is stored on the system and a single record is changed in the database, the entire database file will be included in the incremental backup anyway. With many such large files, an incremental backup will save little, if any, time or space.

Similarly, the frequency with which a backup is performed will be installation-dependent. When a disaster occurs on the system, all information which has changed since the last backup will be lost. So the "safest" solution is to perform backup on a continuous basis. This, however, ties up the system and degrades performance. Many installations perform a backup in the evening after business hours, when the number of users on the system is lowest. With this method, the worst case would entail losing an entire day's work.

Since access to backup information is rare, and often a large portion of the backup is being accessed, random access to backup information is not necessary. So, backups are generally stored on offline serial devices.

> **A device used for storing backup information must provide a cost-effective means of storing large amounts of redundant information. Generally, slower serial devices are adequate.**

2.3 Archival Information

Archival information is the part of the current information that is no longer needed on a regular basis but must be kept around, generally for historical reasons.

- Current information becomes archival information as it becomes outdated. The frequency of this will vary from one installation to the next.
- Archival information is accessed more frequently than backup information, but not on a continual basis—it must be more readily available than backup, but not as instantly accessible as current information.
- Retrieving archival information differs from backup in that it generally entails extracting only a single record or small piece of information from a large file. Therefore, random access to archival information is important.

Archival information includes such things as old versions of operating systems; records on past employees, students, or patients; outdated databases; accounting information from past years; and information on past projects.

The biggest problem with archival information is that when it is accessed, it is typically to restore a single record or piece of information from a large file. For example, a doctor's office may need to access an old patient record, which is stored in a large database file of old patient records. If the information is stored on a serial device, retrieving a single record is a time consuming task, as illustrated later.

Another problem with storing archival information is that the amount of information constantly increases. Unlike current and backup information, which stay relatively constant in size, archival information simply grows.

> **The ideal archival medium offers virtually unlimited capacity that is cost-effective, with moderately fast access to stored information.**

2.4 The Relationship Between Information Types

The relationship between current, backup, and archival information is a dynamic one, dependent upon the particular organization of an installation. The following three examples illustrate the different needs of different organizations:

1. **Storing Student Records**

   In a university setting, there is a need to maintain student records, including information about the student, and current grades and class schedules. During each semester, this information is current. At the end of each semester, the grades and class schedules become historical. The student information remains current until the student graduates, at which time that information also becomes historical.

   In this setting, current information lasts for a fixed, relatively short term. Every semester boundary will guarantee that at least some of the current information in the student records database will become historical. In this case, information could be stored to archives at fixed intervals. Backups may or may not need to be performed on a daily basis as student records, at least by mid-semester, should not change often.

2. **Storing Medical Records**

   Medical records have no fixed turnover of current information. Some conditions require only a single visit, while others may entail months or years of care.

   In this setting, current information changes to archival information every time a treatment ends. There is no way to predict how often this will happen. Some of these will be very short-term, and some will be longer term. Generally, though, some information becomes historical relatively frequently. In this case, information will probably be stored to archives on a periodic, frequent basis. Backups on a daily basis are essential in this case, as new information is added to current patient records.

3. **Storing Employee Records**

   Employee records have variable boundaries, as this information becomes historical as employees leave the company. Hopefully, this is not a frequent occurrence; but it usually cannot be predicted. In this case, information will probably be stored to archives on a periodic, infrequent basis. Again, backup of employee records may not be required on a daily basis.

3 Common Organization of a Data Processing Center

A typical data processing center in the HP 3000 environment is shown in Figure 1.

Figure 1: Typical Data Processing Center

3.1 Equipment Configuration

In this environment, the HP 3000 is connected to terminals and printers, and different storage devices. The types of storage devices that are typically configured into the system can be divided into two categories: fast online storage, and slow offline storage. Fast online storage is provided with system hard disk drives, which offer fast random access to files. The disk drives may be configured as system disks or non-system volume sets (private volumes). Slow offline storage is provided with tape drives. The attributes of each of these types of storage is shown in Table 1.

	Hard Disk	Tape
Capacity	fixed	unlimited
Storage Cost	high	low
Random Access Speed	fast	very slow
Serial Access Speed	fast	moderate

Table 1: Hard Disk and Tape Storage Device Attributes

Hard disks are fast random-access devices, their down side being high cost and fixed capacity—in order to increase storage space, you must add more devices to the system. Tapes, on the other hand, are low cost devices with unlimited capacity—to increase capacity, you simply need to buy additional storage media. Tape drives, however, are extremely slow and limited to serial access.

The drawback to this common data processing configuration is this: the system has two types of devices (fast online, and slow offline), but there are three types of information that need to be stored (current, backup, and archival).

3.2 Accessing Current Information

The essential feature of a device used for storing current information is the ability to quickly and easily access, create, edit, and remove information. The information must be completely and directly accessible to the user. In the common data processing center, access to current information is generally provided with hard disks attached as system or non-system volumes.

This is ideal for current information. The high cost of using hard disks for fast online storage is not a major consideration, as the amount of current information on a system remains fairly constant.

3.3 Storing and Retrieving Backup Information

Devices for backup must be cost-effective. Storage and retrieval is not an issue, as storage is generally done when users are not on the system, and retrieval is rare. In a typical data processing center, backup information is stored serially to tape drives, and the media stored on-site but offline. This is ideal for backup information.

The fact that tape drives are limited to serial access does not adversely affect backup information, since typical access involves restoring an entire system, device, or (less frequently) an entire file. One should never need to access a single record from one file on a backup tape. Reasonble restoration time is expected, but speed is not a major factor. And since a backup operation is replacing corrupt data, there is no need to worry about having sufficient disk space on which to restore the backup—the backup information is simply stored on top of the corrupt data.

3.4 The Culprit: Storing and Retrieving Archival Information

The culprit in this configuration is the archival information. The ideal device for storing archival information is a cost effective device that offers moderately fast random access to information, and connects just like a system disk (with information stored in the same format as the current information, but with unlimited capacity). Neither fast online (system hard disk) nor slow offline (serial) storage in this common data processing organization provides this combination of attributes. For most installations, the solution is to try to fit a square peg into a round hole by "misusing" one or the other of these two storage types.

In one case, archival information is stored along with the current information, on system hard disks. While fast and convenient for those who need access to the archival information, this is a very expensive solution. Since the body of archival information is constantly increasing, more and more system hard disks will need to be added as time goes on. Eventually the limits of the system or the limits of the budget will be pushed: it's a toss-up which one will be exceeded first.

In the other case, archival information is stored offline, like backup, in a serial medium format. While this solution is highly cost effective as far as storage costs, it is very costly in time and effort when information needs to be retrieved. Random access is not required with backup information, since it is typically an entire device or system that is being copied over corrupt current information. With archival information, however, the user typically needs to extract a single record from a large file.

For example, a university may need to access information on a student who attended many years ago. Extracting a single student record from a file containing perhaps thousands of records on students that graduated that same year would be extremely cumbersome. To retrieve this record from an offline serial device, the entire file containing the record would need to be restored before the record could be retrieved. To do this, a "staging" area must be cleared on the system disk, generally requiring the temporary offloading of current information to make room. After the necessary record is retrieved the archive file can be purged from the disk, and the offloaded current data restored onto the system. This time consuming activity can waste an entire day.

- Fast online storage, or hard disk, is ideal for storing current information.
- Slow offline storage, generally serial tape drive, is ideal for storing backup information.
- Neither fast online system storage nor slow offline serial storage has all of the features necessary for an ideal archival medium. This can present a major problem in any data processing environment.

4 Optical Disk Technology

Optical disk technology is quickly increasing in popularity as users discover all that is has to offer: namely, relatively fast random access to a remarkable amount of data, stored on a compact, removable cartridge. Optical disk devices, which use lasers to store and retrieve information from optical disks, were first developed as an alternative to the Video Cassette Recorder. In 1978, the first optical disk system—the Laser-Disk Video Player—appeared on the consumer market. Since then, optical recording technology has been further developed by the mass storage industry, and split into three distinct branches: CD-ROM (Compact Disk Read Only Memory), WORM (Write Once, Read Many), and Rewritable.

4.1 CD-ROM Technology

CD-ROM (Compact Disk Read Only Memory) disks are not generally useful for most mass storage applications, since they are "Read Only" memories: information can be written to the disk only during the manufacturing process, not by an end-user. A master disk is used to duplicate the information by "stamping" the information onto other disks. CD-ROMs are largely used to distribute and reference large amounts of relatively static data such as on-line encyclopedias, legal citations, and (of course) musical recordings.

4.2 WORM Technology

WORM (Write Once, Read Many) optical disk systems store information on a hard plastic cartridge. Unlike CD-ROMs, information can be written to the disk by the end-user, but only once—the writing process causes permanent alteration of the disk surface. WORM optical disk drives write information using a laser which burns pits into raised portions of a spiral track on the surface of the disk. Once a pit has been created, that area of the disk cannot be restored to its normal flat surface: thus information written to a WORM disk is permanent. Figure 2 shows a vertical cross section of a WORM disk.

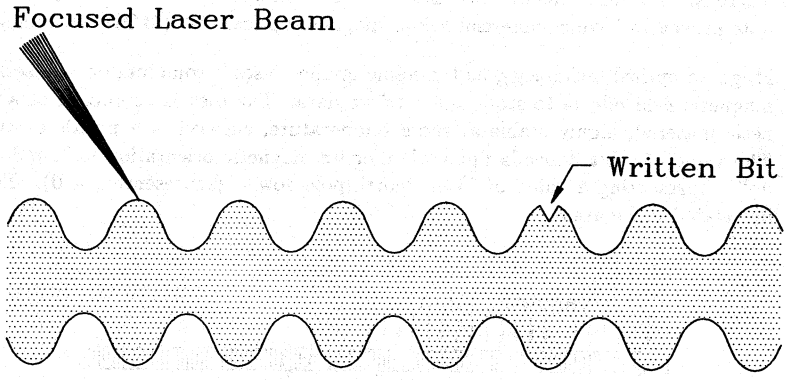


Figure 2: WORM Optical Recording

The same laser beam that is used for writing is used (at a much lower power) to read the information that has been recorded: each pit is interpreted as a digital 1, and each land ("no pit") is interpreted as a digital 0. Pits and lands are identified by the manner in which light is reflected off the surface of the disk.

WORM optical drives have one major "snag" that is not encountered with other mass storage technologies: most existing file systems are structured so that some space on the medium is reserved for a directory. This directory must be updated each time a file is added, edited, or deleted. Since WORM optical disks cannot be rewritten, such directory maintenance is nearly impossible. A number of solutions have been devised to overcome the WORM directory maintenance problem. The successful solutions treat the WORM disk as a serial device when writing, and as a random-access device when reading. These solutions make WORM drives ideal for storing archival information.

Although there are disadvantages imposed by the write-once limitation of WORM drives, they do have one unique advantage. Once data is written, it cannot be altered. This characteristic makes WORM drives excellent for storing information that must be kept for legal and audit considerations.

4.3 Rewritable Optical Technology

Rewritable optical disks are very similar to WORM systems, except that the manner in which information is stored makes it possible to erase and rewrite information. There are several forms of rewritable optical technology, only one of which has to date proven to have commercial value: magneto-optical disk (MOD) technology.

Magneto-optical technology, as the name implies, uses a combination of lasers and magnetic field effects to store and retrieve data. The disk is composed of a magnetic material, highly stable at room temperature, encased in a plastic cartridge. The value of a bit depends upon whether its magnetic orientation is "north-pole-up" (representing a value of 1) or "north-pole-down" (representing a 0). This is illustrated in Figure 3.

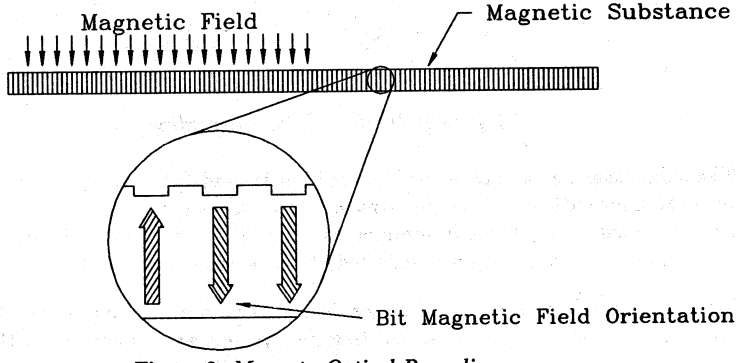


Figure 3: Magneto-Optical Recording

A blank MOD cartridge has all of its bits pointing north-pole-down. A coil in the drive produces a magnetic field that points north-pole-up. The strength of a magnetic field required to change the orientation of a bit varies with temperature: at room temperature, the magnetic coil is too weak to induce such a change. However, at temperatures above 150 degrees Celsius (300 degrees Fahrenheit), the force required to change the magnetic orientation of a bit falls to a manageable level, so bits are easily "flipped" by the magnetic coil.

To write to the disk, a laser heats a spot on the disk to the critical temperature, at which point the magnetic orientation at that spot (representing a bit) can easily be changed by the magnetic field generated by the magnetic coil. After the disk cools—only microseconds later—the magnetic orientation once again becomes nearly impervious to magnetic fields.

To read stored information, an MOD drive directs a low-power laser at the surface of the disk. the light reflected from the surface will rotate in a clockwise or counterclockwise direction depending upon the magnetic orientation of the material at the point of reflection (representing a bit), allowing the data to be interpreted.

4.4 Autochangers

Autochangers, also called jukeboxes, autofeeders, or library systems, use robotics of some sort to swap optical disk cartridges from a large library of disks to one of several online optical drives. Autochangers are used when massive storage capacities and an access time of less than a minute are desired. These systems typically have total storage capacities in the tens to hundreds of gigabytes, with enough internal optical drives to have 5 to 15% of the total capacity mounted and accessible to users at one time.

A home CD player that holds multiple disks and loads one as the previous disk finishes playing is a very simple example of an autochanger. A more pertinent example is Hewlett-Packard's C17*xx* line of Rewritable Optical Disk Library System products. These autochangers come in models that support one, two, or four internal optical drives.

Autochangers, being relatively new to the computing world, are truly useful only when their special needs are taken into account. Existing operating systems have never had to deal with autochangers, and therefore lack sufficient "intelligence" to deal with them effectively on their own. The biggest problem is that existing operating systems have no way of keeping track of what information is stored on which disk in the library—leaving it up to the user to remember where information is stored, and to ask the autochanger to load the correct disk. Specialized tools are needed to keep track of what information is stored where, so that users can simply request a file by name and trust that the autochanger will load and mount the correct disk.

5 Organization of a Data Processing Center Using Optical Technology

Now, look at another possible organization for a Data Processing Center, this time employing optical disk technology:

Figure 4: Proposed Data Processing Center

Once implemented, this organization solves the fundamental problem of dealing with archival information.

5.1 Equipment Configuration

As with the previous configuration, the HP 3000 is connected to terminals and printers, and different storage devices. However in the previous example, three types of information (current, backup, and archival) were being stored on two types of storage devices (fast system online and slow serial offline). With the addition of optical technology, a third type of storage device emerges: we'll call this slow online storage. The attributes of each of these types of storage are shown in Table 2.

	Hard Disk	Tape	Optical
Capacity	fixed	unlimited	unlimited
Storage Cost	high	low	moderate
Random Access Speed	fast	very slow	moderate ($\frac{1}{2}$ to $\frac{1}{4}$ the speed of hard disk)
Serial Access Speed	fast	moderate	moderate ($\frac{1}{2}$ to $\frac{1}{4}$ the speed of hard disk)

Table 2: Hard Disk, Tape, and Optical Storage Device Attributes

Optical drives fall in the middle, between fast online system storage and slow offline serial storage, having unlimited capacity and random access capabilities, but somewhat slower performance than hard disks and higher media costs than tape (but much lower media costs than hard disk). It is this third storage medium that offers the ideal solution to the problem of storing archival information.

5.2 Accessing Current Information

As with the earlier configuration example, access to current information is provided with hard disks attached as system disks or non-system volume sets.

5.3 Storing and Retrieving Backup Information

Backup information is still stored on the low-cost serial tape media which is ideal for this purpose. To optimize the backup process and cost effectiveness, reel-to-reel tape drives can be replaced with the new helical scan tape drives. Both 8mm and 4mm tape drives which employ helical scan technology are available, offering much higher capacities and greater reliability than other traditional tape storage mediums.

5.4 The Solution: Storing and Retrieving Archival Information

While it is the storage of archival information that creates problems, it is in precisely this situation that optical technology has its best fit.

Because an MOD is a random-access device, it can be mounted on the computer just like a system hard disk, allowing information to be stored just like on system hard disk. When the information is no longer needed, the optical disk can simply be removed from the drive and stored on a shelf, at an enormously lower cost per megabyte than hard disks. This tremendous cost savings makes the MOD a much better solution than hard disk for archiving information. It is in the retrieval process that the MOD has its edge over serial storage devices. The average amount of time it takes to access data archived on an MOD is less than 30 seconds, for unmounted disks. For an optical disk that is already mounted, access time is in the milliseconds.

Let's look back to our previous example of the university wanting to retrieve a single student record from the archives. When stored on a serial device, the entire database containing the student records from the appropriate year would need to be restored to disk. Restoring the database to disk may involve offloading current information, unless there is sufficient unused disk space to accommodate the file. Once the file is restored, the record can be retrieved—after which the database file can be removed from the system, and the offloaded current information returned to disk. With the MOD, retrieving the record is simple: just load the optical disk containing the appropriate database, and look up the record as if it were stored on the system hard disk as current information.

Once optical technology is introduced into the data processing environment, the solution can be taken one step further with the use of an autochanger, or library system. The use of an autochanger can increase the efficiency of a center by automating access to the information stored on optical disk cartridges. This reduces human intervention, making the process both faster and more reliable.

5.5 Optical Drives in the HP 3000 Environment

There are two basic requirements for attaching optical disk drives (or any device, for that matter) to any computing system: a compatible hardware connection (interface), and a device driver that can communicate with the drive. For WORM drives there is an additional requirement: some sort of solution to the directory maintenance problem described earlier.

In the HP 3000 environment, two types of interfaces are generally available: HP-IB (Hewlett-Packard Interface Bus) and SCSI (Small Computer Systems Interface). Several device drivers are also available. Table 3 below illustrates the current compatibility of MODs, WORM drives, and autochangers with HP 3000 computers.

	WORM Optical		MOD		Autochangers	
Interface:	HP-IB	SCSI	HP-IB	SCSI	HP-IB	SCSI*
Drivers used:	CS80 or tape	none available	CS80	none available	CS80 and tape	no random access

* HP 3000s can access the autochanger as a serial device for backup only, using Turbo-Store on the MPE-XL and MPE/iX systems.

Table 3: Connectivity of Optical Disks on the HP 3000

5.5.1 Attaching WORM Drives

WORM drives are available for HP 3000 computers using the HP-IB interface; although some 3000 computers have access to the SCSI interface, there are currently no device drivers available to support WORM drives. When the HP-IB interface is used to connect a WORM drive, standard HP drivers can be used to access the drive. However, special software utilities are needed to support directory maintenance on these drives.

5.5.2 Attaching MODs

MPE-V systems do not support the SCSI interface, so magneto-optical disk drives on these systems must be attached via HP-IB. Once attached, the device is accessed using HP's CS-80 disk driver. The MOD can then be installed as a Private Volume, Serial Disk, or Foreign Disk.

Under MPE-XL (MPE/iX), MODs can be attached via HP-IB or SCSI. With the HP-IB interface, the disk uses the CS-80 disk driver and can be attached as a Non-System Volume. While HP does support a SCSI interface under MPE-XL, there are currently no device drivers that support magneto-optical drives, though they may be available in the future.

5.5.3 Attaching Autochangers

HP's magneto-optical library systems (HP C17*xx* products) are equipped ONLY with a SCSI interface. This has caused particular difficulty on HP 3000 computers—the Classic (MPE-V) systems do not have SCSI interfaces or drivers; the Precision Architecture (MPE-XL or MPE/iX) systems, though equipped with SCSI interfaces, treat the library system as a strictly serial device, which severely curtails its effectiveness.

IEM's library system controllers address this issue by providing full random access to the library systems through the HP-IB, rather than SCSI, interface. The internal MOD drives in the library system are made to appear as HP 7935 removable disk drives, allowing them to be installed and operated using standard HP drivers and utilities. The library system picker mechanism itself is also made to appear as a standard HP device, allowing programs to be written for the library system using high-level system file access calls, such as `FOPEN`. Application programs can also be written to manipulate cartridges in the library system.

6 Media Management for Autochangers (Library Systems)

The final step in increasing the efficiency of a data processing center is to "fine-tune" the use of autochangers. While autochangers have provided a big step toward total automation of data processing centers, they have only provided a portion of the total solution. Library systems provide a huge storage capacity, and library system controllers provide the means of automatically loading and unloading disks. However, the entire burden of keeping track of where the information is stored is placed on the user. HP offers library systems that provide up to 93 gigabytes of storage on 144 optical disk—a tremendous amount of information for the user to keep track of. Library systems will reach optimum functionality only when a system is provided to keep track of where information resides on the disks in the library system.

So, the missing link is a media management system. Such a system would, ideally:

- Allow the autochanger to be treated as one gigantic disk;
- Manage (hide) all the details of where files are stored;
- Manage (hide) all the details of file retrieval;
- Manage (hide) all the details of file creation;
- Allow existing applications to run without changes;
- Support multiple users (multitasking).

Access would be expected to be somewhat slow when information is stored on an unmounted disk (15+ seconds), due to the need to swap disks in and out of drives, and the slower access speeds of MODs compared to hard disks. Access speeds would be less than one second for a mounted disk.

When the total media management solution is finally devised, the true value of library systems will be realized, and the idea of a totally automated data processing center will become a reality.

Programming for POSIX Compliance
Planning for Portability

Paper # 7007

Steve Rajavuori
O'PIN SYSTEMS
7900 International Drive
Bloomington, MN 55425
(612)854-3360

Introduction

As "open systems" becomes more of a reality than a buzzword, software developers are beginning to realize the benefits of adhering to the newly developing standards that make systems "open." POSIX (Portable Operating System Interface definition) is a set of IEEE standards that have great significance for developers and users alike: the promise of truly portable software. Using POSIX, developers can conserve time, money and opportunity as they plan for the future by developing applications that will run on varying platforms with little or no change to source code.

This article will discuss the history and rationale for a POSIX-based development effort, and provide the application programmer with a base of understanding to program for POSIX-compliance. You will learn which issues the POSIX standards cover, and which issues are not covered. You will also see how to plan for portability, and learn how MPE/iX implements the POSIX standards. A basic knowledge of "C" programming on either the HP3000 or HP9000 will maximize your benefit from this article.

Background

The information presented here represents a summary of knowledge gained at O'PIN Systems during our experience in developing products that conform to the POSIX standards. O'PIN Systems was founded in 1985, with a vision to create state-of-the-art software for improving the productivity and efficiency of knowledge-workers, managers and other sophisticated users of information in large organizations. O'PIN's MEDSYTE™ decision support system for health data analysis is now used by leading managed care organizations throughout the U.S. REVEAL™, our electronic reporting software, provides information access to HP users throughout the world.

Our development environment during that early time in our history was typical of many HP3000 shops. We used primarily COBOL and Pascal as our programming languages, VPLUS for screens and IMAGE for database management. We geared the user-interfaces for our systems toward an HP terminal.

Over time, however, as the "open systems" movement gathered steam, a new vision began to evolve for our company. We discovered that the management systems we were developing were in reality reshaping the information processing and decision making parameters within our client companies. We recognized that many of our clients needed applications that would faciliitate their own platform independent plans for the future. Their needs to be more globally competitive were responsible for pushing us toward the outer edge of emerging technologies.

Ray Pinson, O'PIN Systems' president, set our direction for new technology development in late 1990. We began making plans for a new generation of products that would be implemented with client-server technology, use graphical user interfaces and conform to new industry standards for portability.

In January 1991, key members of our development team attended the Technology Camp sponsored by Hewlett-Packard. In that forum it became clear to us that our strategy was right on target. "Open systems" was quickly becoming a major focus of interest, not only within the HP community, but in all areas of the computer industry. Standards were taking shape that would influence development for years to come.

We determined that conforming to the POSIX standards for new systems development would provide the best opportunity of having code that could port easily to various systems. We made this decision based on two important factors. First, POSIX has become a widely accepted industry standard for operating system interface. Almost all major vendors are already producing POSIX-compliant operating systems, or plan to in the near future. (The recently announced Unixware and Windows NT systems both claim POSIX-compliance.) Second, HP's decision to make MPE/XL POSIX-compliant helped us, since we were already established in the HP3000 environment.

In 1992 we implemented our plans, working first on the HP9000, then later on the 3000. We developed a system that would operate identically on either machine, with almost no code changes between platforms. We took advantage of the many portable features of POSIX, while discovering a number of system-dependent issues not yet covered by POSIX.

What is POSIX?

The concept of having "open systems" -- computers of any variety that can cooperate and communicate easily -- is becoming both desirable and necessary in many companies. Programs and data from one system must be usable on systems of different sizes and different manufacturers. One of the basic requirements for having open systems is that computers provide a common interface to users. Since the operating system is the most fundamental user interface to a computer, the operating system interface (not the underlying code) must work the same way on every platform.

The POSIX standards represent the latest development in an ongoing effort within the computer industry to create a standard for operating systems. The intent of the POSIX standards is to define how an operating system ought to look to a user

(who, in the context of this article, is a programmer). POSIX defines standards for system functions, shell commands, communication methods and administrative services, plus a number of other, more specialized subjects. Systems that fully implement these standards will run programs and scripts from other compliant systems with little or no change to source code.

POSIX-compliant operating systems still differ -- each operating system is hardware dependent, and may have additional features that go beyond the standards. However, a compliant system assures the user that certain defined services and methods will be available.

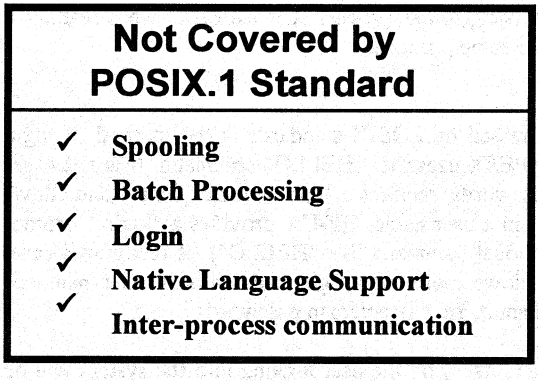
While POSIX is derived from UNIX System V and Berkeley UNIX, it is not an operating system. POSIX does not define how to write application programs or how to write the operating system. Instead, it defines the interface between applications and their libraries.

There are more than 11 different projects making up the family of POSIX standards. This article will focus on the first two: POSIX.1, which defines the library of system calls used to access operating system functions, and POSIX.2, which specifies a shell command language and provides over 70 utilities. Hewlett-Packard implemented these two standards in MPE/iX Release 4.5. (NOTE: There is no plan to provide POSIX support for MPE V.)

What the POSIX standards do not define

There are a number of important items that are not covered by POSIX standards, or are defined in standards that are not yet complete. Programmers must take great care in designing portable systems that deal with these issues, as they are likely to vary between systems. Figure 1 lists issues that are not covered by the POSIX.1 standard.

Figure 1



When a need arises to use some portion of the system that is non-portable or non-standard, the general approach should be to make the outward appearance (user interface) of the application appear uniform across platforms, even though the underlying code may be quite different.

Spooling

POSIX.7 will eventually standardize spool management. Today, however, spooling differs between UNIX systems. Even greater differences are evident when comparing UNIX systems to non-UNIX, POSIX-compliant systems like MPE/iX. Any similarities between the MPE/iX spooler and the HP-UX spooler can only be found at a conceptual level, at best.

Batch (background) processing

Batch, or background, processing is implemented quite differently on various POSIX-compliant systems. HP-UX uses the typical UNIX "cron" facility for starting batch processes, while MPE/iX has many proprietary commands provided for the design and management of batch processes (:STREAM, :JOB, :ABORTJOB, etc.). Programmers familiar with the MPE/iX batch facilities need to be careful with the use of job control commands and functions in their development. Many features available on MPE/iX may not be found on other POSIX systems.

Listing 1 shows an example of how to address a typical background processing issue on two different systems. We needed a daemon process (a continuously running background process) to provide a connection to our application for clients wanting to connect to the server program, through TCP/IP. The server had to be always running, listening for a connection request from a client.

On HP-UX, we used a script to set the necessary environment variables, and then run the server program as a background process. On MPE/iX, we streamed a batch job that accomplished the same purpose.

Login

Another critical item not addressed by POSIX standards is the method of login used by operating systems. MPE/iX uses the ":HELLO" command. It is not case-sensitive for user names or passwords, requires a "user.account" syntax and allows no more than eight characters in a user name. HP-UX provides a "login:" prompt (but does not require an additional command like "HELLO"). It recognizes case, requires only a user name, allows more than eight characters in user name or password and allows non-alphanumeric characters in passwords.

Any application that depends, in itself, on the user logging into the system will be stuck with system dependencies. In this situation, be sure to document the differences between the systems.

Listing 1: Using a background process: MPE/iX vs. HP-UX

HP-UX

For HP-UX, we set up a simple script that sets environment variables, then runs the daemon program as a background process. (Using the "&" at the end of the line indicates that the program should continue running in the background even after the user running it logs off).

```
$cat devscript
export DEV_HOME=/DEV10/BIN
export DEV_DB_HOME=/DEV10/PUB/control
/DEV10/BIN/PROCMGR 2> pmlog  &
```

MPE/iX

On MPE/iX, we set up JCL to accomplish the same purpose as the above script. Since MPE/iX does not yet provide for orphaned processes, we stream this job and leave it continually running. For all practical purposes, this behaves just like the HP-UX daemon.

```
1  !job procmgr,mgr.dev10;outclass=lp,1;pri=ds
2  !file daemonlg;rec=-1,,b,ascii;disc=2100000;save
3  !purge daemonlg
4  !sh.hpbin.sys > *daemonlg -L
5  export DEV_HOME=/DEV10/BIN;
6  export DEV_DB_HOME=/DEV10/PUB/control;
7  /DEV10/BIN/PROCMGR 2> pmlog;
8  exit;
9  !eod
10 !set stdlist=delete
11 !eoj
```

NOTES:

Line 4: Output from the shell is redirected to a byte-stream file, since this is what the shell expects. Normally output would go to the $STDLIST of the job, which is not a byte-stream file.
Line 5: Each shell command is terminated by a semi-colon
Line 9: Shell input must be terminated by an "EOD" command

Native Language Support

No POSIX standard exists yet to address the issue of native language support. Fortunately, the X/Open Portability Guide (another widely accepted operating system standard) provides functions to meet this need. Most large vendors (including Hewlett-Packard) are striving to meet both the POSIX standards and the X/Open standards (XPG 4 is the latest standard published by this group). Most UNIX systems (including HP-UX) already comply with XPG 3, and Hewlett-Packard is investigating making MPE/iX XPG 4-compliant in a future release. Thus, it is reasonably safe to use the X/Open native language functions, and still assume portability.

Native language support is definitely an "open systems" issue, which provides for portability across languages and cultures, as well as machine types. XPG-compliant systems use the "cat" functions to read all text messages from a message catalog, rather than coding text messages (such as user prompts or error messages) into programs. "Catopen" opens a message catalog file, "catgets" reads messages and "catclose" closes the catalog file.

As of Release 4.5, MPE/iX does not yet provide the XPG catalog functions. It does, however, provide the same services through very similar MPE intrinsics: CATOPEN, CATREAD and CATCLOSE. Listing 2 shows a function that uses both the XPG "catgets" function and the MPE "CATREAD" intrinsic.

Inter-process communication (IPC)

The POSIX.4 standard will define methods for inter-process communication. POSIX.1 provides pipes and FIFOs, but these may not meet all application IPC needs. (MPE/iX does not yet have pipes and FIFOs, but it will in Release 4.7.) Hewlett-Packard decided to provide more UNIX-style IPC capabilities with MPE/iX 4.5 by including facilities known as the "SVID IPC library" (defined by the AT&T System V Interface Definition). These facilities include message queues, shared memory and semaphores. HP also provided another commonly used means of IPC known as Berkeley Sockets (first released in MPE/iX 4.0). Berkeley Sockets are not part of the POSIX standard, but are recognized as a widely used means of IPC, and thus considered important for inclusion in MPE/iX. (HP-UX provides both the SVID IPC library and Berkeley Sockets.)

Listing 2: Using the Native Language Support functions: MPE/iX vs. HP-UX

```
/************************************************ defines */
#define _POSIX_SOURCE
#define _XOPEN_SOURCE
#define DEV_CAT  "msgcat"
#define MAX_MESSAGE_SIZE 256
/************************************************ includes */
#include <stdio.h>
#include <errno.h>

#ifdef MPE_ONLY
#  include <mpe.h>
#else
#  include <nl_types.h>
#endif

#ifdef MPE_ONLY
#pragma intrinsic CATOPEN
#pragma intrinsic CATCLOSE
#pragma intrinsic CATREAD

typedef struct {
         short msg_stat;
         short other;
} CAT_STAT;
#endif  /* MPE_ONLY */

/************************************************ globals */
#ifdef MPE_ONLY
int     cat_fd = -1; /* message catalog file descriptor */
#else
nl_catd cat_fd = -1; /* message catalog file descriptor */
#endif

/************************************************ msgcatopen */
int msgcatOpen(void)
{
#ifdef MPE_ONLY
   CAT_STAT cat_stat;
#endif

#ifdef MPE_ONLY
      if ((cat_fd = CATOPEN(DEV_CAT, &cat_stat)) < 1) {
         /* error handling occurs here */
         return 1;
      }
#else  /* XPG3 */
      if ((cat_fd = catopen(DEV_CAT,0)) == -1) {
         /* error handling occurs here */
         return 1;
      }
#endif
   return 0;
} /* msgcatOpen */
```

```
/************************************************ msgcatread */
/* Gets message from 'set', number 'msg' in DEV_CAT.          */
/* Returns pointer to buffer containg message. If catgets     */
/* call fails, return points to default message.              */
/************************************************ msgcatRead */
char *msgcatRead(int set,int msg)
{
#ifdef MPE_ONLY
        CAT_STAT    cat_stat;
        int         size = MAX_MESSAGE_SIZE;
        static char msgbuf[MAX_MESSAGE_SIZE+1];
        int         length = 0;
#else  /* XPG3 */
        char        *temp_chp = NULL;
        static char default_msg[MAX_MESSAGE_SIZE] =
           "Error occurred reading message from catalog";
#endif

   if (cat_fd == -1)
      if (msgcatOpen()) {
          /* open failed; do error handling */
      }

#ifdef MPE_ONLY
   length=CATREAD(cat_fd, set, msg, &cat_stat, msgbuf, size,
            NULL,NULL,NULL,NULL,NULL);
   msgbuf[length]='\0'; /* MPE doesn't provide terminating Null */

   if (cat_stat.msg_stat != 0) {
      /* error handling */
   }
   return(msgbuf);
#else  /* XPG3 */

   errno = 0;

   temp_chp = catgets(cat_fd,set,msg,default_msg);

   if (errno != 0) {
      /* error handling */
   }
   return(temp_chp);
#endif

}  /* msgcatRead */
```

Dealing With Non-Standard Portions of Source Code

When the need arose to use system-dependent functions (such as MPE intrinsics), we generally surrounded the system-dependent portion of code with an "#ifdef/#endif" block, so that only the portion intended for a given system would compile on that system. We used the constant "MPE_ONLY" to indicate a portion of code that would only be compiled on the HP3000. (See Listing 2.)

On several occasions we needed to use a POSIX function that wasn't available yet on MPE/iX. In at least two cases, we were able to obtain (from HP) a fairly simple piece of code that provided the same features as the missing system function (the access() and getlogin() functions). Listing 3 shows the MPE/iX implementation of the POSIX "getlogin" function.

POSIX programming language: C

Future standards will define ADA and FORTRAN bindings to the POSIX standards. For now, C is the language of choice. It was used to develop UNIX and is a standardized, portable language. With the POSIX.1 library functions defined in terms of the C language, C is the only real choice for POSIX-compliant programming.

Compiling Under MPE/iX

Use the "c89" command available in the POSIX shell to compile programs using the POSIX.1 library functions on the 3000. This is actually an interface to the standard C compiler and Link Editor, providing both compilation and linking. The MPE/iX implementation of this command is almost identical to the one provided by HP-UX. "c89" allows the use of HFS (Hierarchical File System) names for any of its parameters, and expects adherence to ANSI C standards. (NOTE: to include POSIX features in your program, such as primitive variable types, you must define the constant _POSIX_SOURCE. The constant must be defined before any system headers are included, so it is usually best to define it in the first few lines of your source file.)

Listing 3: MPE/iX Implementation of getlogin() function

```
/*
   Returns a pointer to a string containing the user's login name,
   or returns NULL if the login name cannot be found. The result
   is in the form "user.account".

   Error codes: none
*/

#define _POSIX_SOURCE

#include <unistd.h>        /* specified by getlogin() */
#include <errno.h>         /* errno and error constants */
#include <mpe.h>           /* ccode(), CCE */

/*****************************/
/*   general information   */
/*****************************/

#define END_OF_STRING_CHAR  '\0'

/************************/
/*   MPE information   */
/************************/

#pragma intrinsic   WHO

#define MPE_NAME_LEN   8
#define MPE_LOGIN_LEN  (MPE_NAME_LEN+1+MPE_NAME_LEN)

char *getlogin(void)
{
   static char login[MPE_LOGIN_LEN+1] = "";
   char        user_name[MPE_NAME_LEN+1];
   char        acct_name[MPE_NAME_LEN+1];
   int         i,l;

   /* get user's login name */
   WHO(NULL, NULL, NULL, user_name, NULL, acct_name);

   /* WHO does not return any status */
   if (user_name[0] == ' ')
      return (NULL);

   for (i=0,l=0; i<MPE_NAME_LEN, user_name[i] != ' '; i++,l++)
      login[l] = user_name[i];
   login[l++] = '.';
   for (i=0; i<MPE_NAME_LEN, acct_name[i] != ' '; i++,l++)
      login[l] = acct_name[i];
   login[l] = END_OF_STRING_CHAR;

   return (login);
} /* getlogin() */
```

It is important to note that a C program compiled by "c89" on the 3000 will not perform quite the same as one compiled outside the POSIX shell. "c89" links with the POSIX libraries, while programs linked outside the shell are linked with the ANSI C libraries.

File system functions highlight the most evident differences between compiling inside the shell versus compiling outside the shell. Programs which call "fopen()" will create a byte-stream file when compiled by"c89," while the same program will create a 256-byte, fixed-length-record file by default when compiled outside the shell. HP wanted to make sure that the POSIX functions comply strictly with the POSIX standards (which use byte-stream, not fixed-length-record files), while the standard HP C functions provide traditional HP-style files for the convenience of MPE programmers.

The POSIX file system

POSIX defines standards describing the file system for an operating system. The implementation of this standard (which is based on System V and Berkeley UNIX file systems) on the HP3000 is known as the "Hierarchical File System." HFS is significantly different from MPE's native file system. While MPE provides a flat "directory" structure consisting of accounts and groups, POSIX allows a "tree-style" hierarchical directory structure, with a virtually unlimited depth of directories. Filenames may be up to 14 characters long (actually longer, but no more than 14 for portability), and are case sensitive. They may contain the characters period, underscore and hyphen, as well as numbers and letters.

POSIX uses "permissions" to define file security. Permissions are comparable to, but different from, MPE account/group/creator access control. POSIX defines three types of users of a file: the owner, other users belonging to the same "group" as the owner and all other users. Three types of permissions are provided: read, write and execute. Each of the user types may be allowed each of these permissions, making a total of nine permission options for a file.

Directories are also assigned permissions, although the actual meaning of each permission is different for files than it is for directories. Figure 2 explains the use of the permissions for both files and directories. For a more detailed description of the HFS on MPE/iX, please refer to "New Features of MPE/iX 4.5: Using the Hierarchical File System" (HP Part No. 32650-90351).

Figure 2

	Read (r)	Write (w)	Execute (x)
File	May read file	May update file	May execute file (script or program)
Directory	May list files in directory	May save, rename or delete files in directory	May "cd" into this directory (to make current working directory

The POSIX shell

The working environment for a POSIX programmer is the POSIX shell. Based on the Bourne and Korn shells of UNIX, the POSIX shell is defined by the POSIX.2 standard. The shell standardizes commands, environment and user utilities. This allows programmers to easily move from one system to another without having to relearn a new operating system. A UNIX programmer working in the POSIX shell on the 3000 will feel quite at home.

While the POSIX shell does not provide some of the more complex features found on some UNIX systems, it does provide many frequently used services. The basic shell commands dealing with files and directories (ls, rm, mkdir, etc.) are very similar between HP-UX and the 3000 POSIX shell. The shell also provides popular utilities such as grep, awk and vi. On the MPE/iX POSIX shell, there is even a built-in e-mail system (mailx) -- something new for 3000 users.

The normal way to run the POSIX shell on MPE is to run SH.HPBIN.SYS with the info string "-L." The shell then becomes your new command interpreter and, for all appearances, you are on a UNIX (POSIX) system. The "callci" shell command provides access to MPE/iX commands.

MPE/iX issues

While MPE/iX is rapidly becoming a POSIX-compliant operating system, it is not yet complete. HP has done a commendable job in providing a significant portion of the POSIX.1 and POSIX.2 features. Future releases should make MPE/iX an even better POSIX implementation.

Programs in Groups

MPE/iX Release 4.5 is the first release to include the POSIX.1 library, the POSIX.2 shell and the Hierarchical File System. Programmers planning for portability need to look carefully at the MPE/iX POSIX manuals to discover which functions and features are not yet implemented. One of the more significant restrictions of MPE/iX is that programs must be stored in MPE/iX groups -- not HFS directories. Attempting to run a program stored in a directory will cause an error. In the POSIX shell, the error message is "Implementation-defined error." If attempted from CI, the message is "HFS names not supported by this parameter. (CIERR 192)."

Byte-stream vs. Fixed-length-record

Another issue to watch out for is the difference between byte-stream files and standard MPE fixed-length-record files. POSIX utilities cannot read fixed-length-record files. Attempting an operation such as "cat" or "vi" on a fixed-length-record file will result in an "Implementation-defined error." Fortunately, the MPE/iX POSIX shell provides two utilities to convert files from MPE-style to byte-stream ("tobyte") and vice-versa ("frombyte").

Warning: "frombyte" has no parameter for record size -- it only creates files with 80-byte records. If you have a source file with lines longer than 80 characters, the lines will be split in two by frombyte. (Use FCOPY as an alternative in this case.)

STORE

Currently, to include HFS files in a full backup on MPE/iX, you must use "/," rather than "@.@.@." "@.@.@" will not include files in directories, or with HFS names. This approach has been criticized widely, and in response, HP will make "@.@.@" include ALL files (including those with HFS names) in Release 4.7.

Debugging

Currently there is no symbolic debugger available for programs compiled by "c89." POSIX programmers on the 3000 need to plan on using plenty of "printf" or "fprintf" calls to debug.

The guidelines

There are several good sources for guidelines on POSIX programming. For a thorough, very readable description of the POSIX.1 standard functions and features, see "POSIX Programmer's Guide," by Donald Lewine (O'Reilly & Associates, Inc., 1991). This book explains portability issues very clearly, as well as including a complete catalog of all the POSIX.1 functions.

The "MPE/iX Developer's Kit" (product number 36430A) is required to compile with POSIX.1 functions on the 3000. This provides you with several manuals that describe the POSIX.1 functions under MPE/iX, as well as the C compiler and POSIX.1 libraries needed to write POSIX programs for the 3000.

Another very helpful source of information is the "man" pages on HP-UX. We consulted the "man" page every time we wanted to see the standards conformance for a library function we wanted to use. With very few exceptions, we only used functions that listed either "ANSI C" or "POSIX.1" as standards. See Listing 4 for an example of a library function "man" page.

Limits

The system include file <limits.h> contains a number of limits that are important to observe. Maximum path size, maximum number of open files and maximum number of child processes are just a few of the items defined here. To be truly compliant, an application must not exceed these limits. Unfortunately, some of the limits are fairly small. If you choose to exceed the limits allowed by POSIX, be careful to document the exceptions and be prepared to make changes when porting to other systems.

Variable Types

POSIX defines a number of primitive variable types for function arguments. These are defined in the system include files, and end with the suffix "_t." For example, all functions that deal with processes use the type "pid_t" for process ID. This is usually implemented as an "int," but a program should always declare variables for these functions as "pid_t," rather than "int." This ensures portability if a particular system was to use a different type for its implementation.

Listing 4: HP-UX man page for POSIX Library Function

```
getlogin(3C)                                          getlogin(3C)

NAME
      getlogin - get login name

SYNOPSIS
      #include <unistd.h>

      char *getlogin(void);

DESCRIPTION
            .
            .
            .
ERRORS
      getlogin fails if any of the following is true:

           [EBADF]      An invalid file descriptor was obtained.

           [EMFILE]     Too many file descriptors are in use by
                        this process.

           [ENFILE]     The system file table is full.

FILES
      /etc/utmp

SEE ALSO
      getgrent(3C), getpwent(3C), cuserid(3S), utmp(4).

DIAGNOSTICS
      getlogin returns the NULL pointer if name is not found.

WARNINGS
      Return values point to static data whose content is
      overwritten by each call.

STANDARDS CONFORMANCE
      getlogin: SVID2, XPG2, XPG3, POSIX.1, FIPS 151-1

Hewlett-Packard Company    - 1 -    HP-UX Release 8.0: January 1991
```

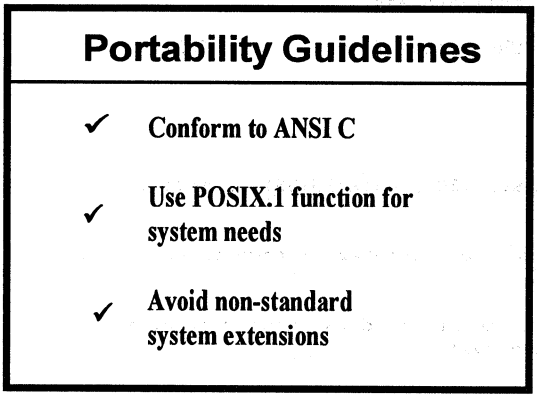
Notice the last section -- "STANDARDS CONFORMANCE". Both POSIX.1 and FIPS 151-1 (a subset of the POSIX.1 standard) are listed for this function, indicating that it is highly portable.

Can an application be completely portable?

Many HP sites are currently evaluating the addition of UNIX operating systems to their inventory of computer systems. As that happens, the need for greater portability will increase dramatically. Our mission, for our company and for our clients, is to continue to play a major role in facilitating the creation of truly open systems by concentrating on providing maximum portability. This will enhance the ability of our customers to communicate quickly and easily across platforms and across continents. And software developers will be the key architects in this wave of activity.

The real question for developers, though, is this: Can you create a sophisticated system that can be recompiled with no source code changes on the 3000 and 9000? On other non-HP systems? The answer is yes, if you follow the guidelines and stick to the rules: conform to ANSI C, use POSIX.1 functions for system needs and avoid non-standard system extensions. (Summary in Figure 3.)

Figure 3



The acceptance of POSIX as the standard for operating systems provides the foundation for developing portable applications. Operating system vendors appear to be willing to make POSIX a cornerstone for future plans. Now it is up to us -- the application developers -- to take advantage of the benefits of portability, by making our programs meet the standards.

Paper # 7008

Title: HP's Use of Business Workstations

Andrew J. Phillips
Technology Solutions Lab
2015 South Park Place
Atlanta GA 30339
(404) 850-2937

Introduction

This article is intended to provide information and insight to those already using or contemplating using UNIX workstations in a business environment. Hewlett-Packard engineers have used workstations since they were invented; and in the beginning, this use was mostly limited to technical professionals requiring significant processing power to run their applications (such as CAD/CAM). More recently, "the workstation" has infiltrated markets that didn't even exist 20 years ago. This article is about how that infiltration happened at a part of Hewlett-Packard. We found that, with the proper precautions, we could harness the power of the workstation for the average service or administrative worker.

I have structured this article such that the first part is concerned with the major decisions made in implementing workstations, and the latter part goes more into the technical details. Everyone reading this should at least skim the first part because the overall strategy is discussed, and latter parts may seem unneccesarily complex without the "big picture".

SOME DEFINITIONS AND TERMINOLOGY

The first order of business is to answer the question " What is a Workstation anyway ?", so we can all be assured we are talking about the same thing. For this article, we will define a "workstation" as an HP9000 (any series) with graphics hardware running HP-UX (any version above 6.5) and the X window system, See [1] for a better description. Although this definition is rather broad, it will suffice for the kinds of things we will be talking about. The

hardware actually used in this case study is mostly Model 425s, some 700s and a smaller number of older models like the 340 or 360.

Since this article is based on real life experiences, the next order of business is to define the context of this article, that is, who are we talking about ? "Who" in this case, is Hewlett-Packard's North American Field Operation (NAFO) which includes the Atlanta Campus (Response Center, Atlanta ITC and Riverwood Bldg.), and all the field sales offices in the United States. I will also refer periodically to "IT", meaning the Information Technology department which is responsible for running the data center, supporting applications, and installing and supporting computer hardware and related technologies.

Lastly, again for clarity, I want to define some frequently used terms. A display is the CRT screen that sits on a users' desk. A window is an object that appears on a display. For the purpose of this article, almost all the windows will be rectangular in shape and look like an Hewlett-Packard display terminal screen - 24 lines by 80 columns. An application is what appears inside a window. A Graphical User Interface (GUI) is typically a bitmapped display (the graphical part) using a windowing system (in this case X11) that provides the user a means for managing the windows. Hewlett-Packard VUE is another example of a GUI, even though VUE really uses the X window system (X11). Also, hpterm (see the man page if you have an HP-UX system) is the terminal emulator used to access applications, and vt3k (again, see man page) is the datacommunications program used to login to remote HP3000s across the network. To help differentiate between different types of end users, I will use OLTP (online transaction processing) for those who use primarily production aplications on an HP3000. It is this type of user that is dealt with in this article. That is not to say that the ideas described here apply only to this kind of user, but that this particular case study specifically addresses the needs of this kind of worker.

BRIEF HISTORY

In 1989, Hewlett-Packard installed about 400 workstations in the response centers. This was a risky project, and people bet their careers on this new paradigm and that people such as call coordinators and response center engineers would be savvy enough to use a mouse and windowing environment and that the existing HP3000s wouldn't become too overloaded with the extra sessions and processing load. Well, this gamble obviously worked out in favor of workstations, but there were some problems related to integrating with existing applications (see PITFALLS section).

In late 1990, another 400 or so workstations were installed in Europe to act as multi-window displays for existing HP3000 applications, with a documented 12 % increase in productivity. Mostly due to the elimination of the need to log off and on to other systems. These encouraging results sparked various "pilot" projects throughout the company that further demonstrated the potential, but usually at a high administrative cost. Something had to be done to allow using departments to install workstations and still concentrate on the primary activity of helping customers and not the technology itself.

In 1991, a task force (consisting of many of the early implementors) was formed to consolidate and leverage experiences with workstations in the business environment. This resulted in the Workstation Best Practices document, a set of guidelines on how to set up, configure and administer workstations for Hewlett-Packard's internal business functions, distilled from the lessons learned at many pilot sites.

BEST PRACTICES

The Best Practices document was written to meet this end. They analyzed nearly a dozen completed or nearly completed workstation projects. The review team was made up of the experts that were directly involved in these projects. The results of this analysis were organized to help business managers considering workstation purchases answer the same questions that you probably have, specifically:

1. What feature and benefits might my organization receive from implementation of workstations?

2. What are the obvious (capital and expense dollars) and hidden costs (ongoing support and ongoing investments) of deploying this technology?

3. Will workstations help my department? Are they the right solution? How about PCs?

4. Once I have decided to proceed, exactly what should I order and how should it be configured for ease of use and support.

As stated earlier, most of these projects were targeted towards online transaction processing (OLTP) environments. It is in these environments that immediate gains and hard dollar savings can be realized. Since this article draws on the knowledge gained from these pilot installations, the major focus is toward OLTP. This is not to say that workstation are only for OLTP uses or that the ideas and suggestions presented here only apply to this environment. The reader is encouraged to be creative when thinking of other areas to implement this technology.

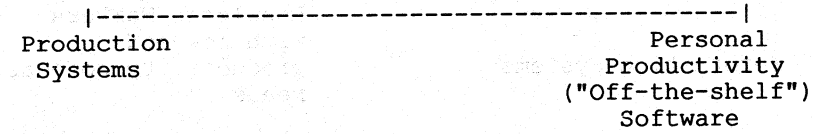
In general, workstations were found to be of greatest benefit to those users that spend most of their computing time accessing production systems while PCs are most useful to personal productivity users.

It is also worthy of note that the best practices task force delivered a "starter kit" for sites wishing to implement workstations that included a set of templates to set up a standard environment, along with a set of utilities. See the SOFTWARE CONFIGURATION section for more about the starter kit.

SHOULD MY GROUP USE PCs OR WORKSTATIONS ?

Determining the roles of the PC and the workstation requires an examination of both the benefits of the two platforms and the tasks conducted by users. For some users it is not clear which is the best client at this time. But for many users, the answer is clear. In areas where client positioning is not clear, efforts are being made to bridge the gaps in functionality using emulation software and by negotiating with vendors to provide software that is consistent across both platforms.

Since it is impossible to reduce user needs into a simple set of categories, a continuum can be employed to help describe user needs. At one end of the continuum are personal productivity tools traditionally thought of as PC software. This "off-the-shelf" productivity software includes word processors, spreadsheets, and business graphics programs. At the other end of the continuum are the internally written production systems that store and process the company's critical business data. These systems have traditionally been implemented using a host/terminal architecture, and are typically HP 3000 based.



Users are represented on this continuum based on the time they use or the needs they have for the software at the ends of the continuum. A user who only uses Personal Productivity Software is at one end, a user who only uses Production Systems is at the other, and a user with a mixed set of needs is somewhere in between.

In general, it can be said that users near the Personal Productivity Software end of the continuum are best suited for PCs while users at the Production Systems end of the continuum are best suited for UNIX workstations. While it is unlikely that users' needs will actually place them at one extreme of the continuum, solutions for accessing and running software on either platforms are available for both the PC and the workstation. In particular, software traditionally found on the PC (Productivity Software) is being developed/ported to the UNIX workstation platform.

An additional consideration before positioning client hardware for a user is the use of information access and decision support tools. These tools provide read-only access to company data, processed by Production Systems. While this functionality is available using HP 3000-based tools, many new tools

are currently available for the PC. Users of this new software may feel that a workstation gives them better access to current Production Systems, but these tools require a PC to run.

The table below summarizes these key points of comparison.

UNIX Workstations	PC
Production System Users	Knowledge Workers
Production System Supporters	Users of PC-based
Production System Developers	information access tools and Executive Knowledge Workers with low-end
Information Systems	productivity software needs

The above chart does not specifically address users with highly mixed needs for Personal Productivity Software and Production Systems. These users need to evaluate the pros and cons of using the PC or the UNIX workstation. Some specific recommendations are:

* Where a user access ONLY production systems and normally manages a large number of systems and or applications, a workstation can provide almost immediate benefits. This is especially true when the user is in a "customer on the phone" situation.

* PCs should be used where users have a need for high-end personal productivity tools and/or a need for PC-based information access or decision support tools.

* Users with needs that would justify both a PC and a UNIX workstation (e.g., both high-end personal productivity tools and access to production systems) may need to have two client machines. Where the cost of two machines is prohibitive or where the office space is not available, emulation products such as SoftPC and PC-based X emulators can be used on the UNIX workstation or PC respectively to access software not available for a user's particular hardware.

Workstation users will enjoy a large display full of multiple HP 3000 applications, and will have the native platform for future client/server based production

systems. If their personal productivity software needs are limited to simple text processing and/or spreadsheets, VED and/or Lotus 1-2-3 for HP-UX will meet their needs. If however, they require additional personal productivity software, they will have to turn to interim solutions offering limited data sharing with PC users, and will later need to migrate to recommended software as it becomes available.

RESOURCE REQUIREMENTS

Workstations require system resources to function. Careful thought and planning is needed to determine exactly what resources will be needed. If the application in question in running on a heavily loaded HP3000 series 70, don't add workstations, it'll only make a bad situation worse. The cost of moving to workstations may require the purchase of an MPE-XL machine. All such costs must be considered in a return on investment analysis. Please see the HP3000 IMPACT section for a technical discussion of this topic.

Besides the hardware costs, adding workstations will require additional IT human and LAN resources, hence an increased IT bill. These additional costs, hardware and human should be offset by increased end user productivity. If this can not be demonstrated in project planning, the conversion to workstations should be re-examined.

PITFALLS TO AVOID

In order to provide a realistic picture, I am including this section on items that were troublesome for us. This list is certainly not all-inclusive, and your site may have a different set of requirements or even different problems. However, for whatever it is worth, here are some of the "pitfalls" NAFO encountered.

Lack of resources can really cause trouble. See the section on HP3000 IMPACT for an in-depth discussion about this issue. Another issue was slave printing. We had several applications that made use of slave printers. In fact, two different applications can access slave printers in two different ways. This was initially quite troublesome since the hpterm

emulator did not provide the complete spectrum of this slave print functionality, and we had to find alternative methods. Another issue relating to the hpterm emulator was the fact that some applications did not look exactly the same on a workstation. Specifically those that used the line drawing character set and required a certain termtype were troublesome. See the example resource file for hpterm near the end for more detail on this. Another issue was that while using these windows they could be resized by the user, creating problems for applications that expected an 80 column by 24 line screen. Again, see the section on resource files for a method to avoid this problem.

Also, compatability and differences between versions of software occasionally caused some problems. Another issue was funiture, of all things. We found that the 19" monitors that were recommended simply did not fit some of the modular furniture already in use. Staffing for administration and support can be a problem as well, and remote support can be even more difficult. It is also very worthwhile to make plans that deal with such issues as security, backups, and tape management. I explain how NAFO addressed these issues here, but keep in mind that these solutions are not a "cure-all", and your site may have different requirements.

HOW ARE WORKSTATION USED ?

The basic and primary use for these workstations was and still is to manage multiple HP3000 applications across multiple machines in different cities on a single display. As of march 1993, our data shows that while all of these workstations were using terminal access to multiple hosts (mostly HP3000s), only 8% were running some applications that made use of some local processing power, and another 8% were used in a true client-server environment. This data is from a significant sample, including some 2200 workstations on the Hewlett-Packard internet, with more than 128 servers (both 300/400 and 700 Series). The groups using workstations include credit, telesales, order processing, service contracts, the response center, and of course the information technology department itself. This indicates that a wide variety of business

functions can benfit markedly from the application of workstation technology to managing already existing applications.

HOW WERE THE PRODUCTIVITY GAINS ACHIEVED AND HOW WERE THEY MEASURED ?

The biggest gains were from the combination of having automated access to multiple applications on multiple hosts and using the "cut and paste" function to copy text between applications (see [1] for more specific information about cut and paste). For example, many people within the organization needed to access several applications during the course of the day ... one for e-mail, one for hardware pricing, one for service contracts, etc. Using only a terminal, this process could be described as logon to system1, run application1 ... then logoff and system1 and logon to system2, run application2 and so on. After buying a work-station, this process was reduced to click a button for application1 on system1, then click a button for application2 on system2, etc. essentially eliminating the previous process entirely. This was viewed as a tremendous benefit and enabler to those people with customers on the phone, since now they had "instant" access to all of the applications they needed. In addition, once an application was running, one never needed to close the window and logout unless the system went down for backups or maintenance (or the application timed out). It is easy to see how much of a time savings can accumulate here, especially for those who use a large number of applications.

I mentioned that the cut and paste feature was very well received by the user community, especially those whose jobs involved moving data between applications. This cut and paste feature is not specific to Hewlett-Packard VUE, but a part of the X Window System (see [1]). One uses the mouse to highlight the text to be duplicated in the source window, and then a button click within the target window, and the text is copied. This process is extremely easy to use, and we had people who had never used a mouse or windows before working comfortably in this environment with only a couple of hours of training. Even more importantly, the combination of many applications on one screen along with the cut and paste provided

functionality that was not available at all before. This provided the capability, for instance, for a user to copy a data entry screen into HPDESK for Email and FAXing to customers. This process took 15 minutes before, and now takes less than a minute. If you consider that this activity can take place several thousand times a day, it's easy to see how the benefits can add up.

There was also significant positive feedback about SharedX (see [7]). SharedX allows two or more workstations to "share" the same window. This was used extensively during training sessions where a process was demonstrated to the students with the same window appearing on everyone's display, and then, having seen how it should work, the students performed a similar process individually. Another area that was made easier by the use of this technology was troubleshooting. Administrators could "pop up" a window on a users display and walk them through the process that was giving them trouble, and sometimes identify problems without ever leaving their desk.

There were other minor gains from using such utilities as datebook (a calendar program), calculators, screen editors and other X11 based utilities, but they were not significant compared to the gains from the aforementioned functions. These utilities come "bundled" with Hewlett-Packard VUE, so there is no incremental cost for providing them other than support.

Methods or metrics that were used to determine productivity gains included the number of calls resolved per person per day, the average time taken to close a call, the average time to deliver a quote and the like. Using these metrics, productivity gains of 10% or better were demonstrated within weeks of installation.

An excellent example of the enabling aspect of workstations was the GEM (group enhanced management) project in Park Ridge NJ. There, a team dedicated to specific customers was designed to handle any call that came in, regardless of the question. This technique would not have been feasible without the workstations.

HOW WAS THIS ACCOMPLISHED ?

To provide leadership and focus within NAFO, a project team was formed that included the process owners as well as members of the IT organization. They (see BEST PRACTICES) decided upon the exclusive use of discless clients, and also provided for "cloning" the servers using a customized file system structure and magneto-optical discs (see Technical aspects section). This provided a cookie-cutter process where all the installations had the same hardware, the same software and a very similar file system structure. This similarity between servers also allowed the system management team to automate the cloning process to the point that 2 to 3 different file systems could be generated in a single day. The project team also leveraged heavily off of Hewlett-Packard PSO/SSO services for installing the LAN infrastructure where required and also for the configuration and testing of the clients and servers.

IT's role in this was well-defined from the outset. IT provided guidelines for hardware selection, and crafted a "common operating environment" (COE) where the end users were shielded completely from HP-UX. IT also performed a line capacity analysis to ensure that there would be enough bandwidth between the users and their applications.

Another key component was the "glue" that provided a bridge between the new workstation technology and the existing processes and methods. For example, when users complained that they had no way to tell whether caps lock was on or off, a button was included in the frontpanel that indicated this. Another example was the creation of a program that helped automate the process of making changes to the user environment. While these examples may seem relatively minor, I feel that this "customizing" of the environment to match the business processes is one of the keys to success, and that it will pay off to have some experts ready and able to create this "glue" when needed.

HOW IS THIS SUPPORTED ?

First off, note that support for this environment was shared by several different groups, and also done remotely for the majority of the sites. The groups involved incuded Network services, office technologies, application support for the HP3000 applications and system management. There were also groups that did new implementations and research, see the HOW WAS THIS ACCOMPLISHED section for more details.

IT provided end user training and training for the VUE administrator,who was typically a process expert and part of the remote workgroup, and who was responsible for the "look and feel" of his or her workgroup's displays. IT also defined the support structure (see Figure 3) which placed first line support within the user community, making IT a secondlevel resource. Once the workgroup was up and running, all management of the cluster was done from Atlanta, and the file system was backed up using a local DAT tape drive. It was the responsibility of the VUE administrator or another local person to change tapes daily. Today, the IT Helpdesk takes all calls for support, and routes them to either System management, Office Technology, or application support group.

Using the support structure shown in Figure 3, IT is able to achieve excellent results with only one system manager for every 400 workstations, or one system manager for every 25 servers. There is also only a single VUE support person located in Atlanta, who provides second level support for thirty part-time VUE administrators (equivalent to about four full time employees) in the using organizations. With this hierarchy, the end user has a local contact in their department. We ask that the end user direct his questions and problems to the local VUE administrator. This person may or may not have anyprevious experience with HP-UX, VUE, or even a windowing environment.By having the local VUE administrator coordinate service requests,recurring problems can be handled in a more timely fashion. When a VUE administrator is unable to solve a Vue problem, that person calls the helpdesk. The helpdesk then routes them to the Vue support person in the Office Technology Group.

The support hierarchy is also used to simplify the VUE configuration process. We have mirrored our support hierarchy to the file system hierarchy. The file system hierarchy allows VUE configurations to be made at many different levels. This can be seen in Figure 4.

In this environment, user configurations take precedence over department configurations, etc. Setting up a configuration hierarchy helps prevent duplication of effort, and minimizes configuration errors.

IT manages the NAFO configuration area. This area tends to be a repository of actions and scripts to be used in the department area. The department area is managed by the local VUE administrator. Each department determines what applications they want to use, and if an action is already available in the NAFO area, it is used. If there is not a preconfigured action available, the VUE administrator adds the functionality themselves, often by using the f.exec feature (see the section on AUTO LOGINS). It is to IT's advantage to have actions available in the NAFO area since this helps reduce syntax errors and the level of knowledge necessary at the local level. Occasionally, it is necessary for a user to manage his own configuration, but the department configuration is generally satisfactory.

TRAINING

One final area that must be considered in remote VUE support is training. Again, the support hierarchy used plays a role. IT trains the local VUE administrator in the use and configuration of VUE. It is the responsibility of this administrator to train the users in his department on the use of VUE. The VUE administrator is trained shortly before the workstations are received. Any further training needed by that person due to updates or new applications is done via teleconference and SharedX.

The only problem encountered with this method has been turn over of local VUE administrators. When a VUE administrator leaves the department, a replacement may not be selected, or IT may not

be notified. This causes problems with training, configuration, and support. We are now working with our business partners to remedy this.

STRATEGY FOR THE FUTURE

IT will again meet with the process owners to assess the situation, and to refine the processes and decide upon improvements. We know that there will be more client-server applications coming in 1994 and 1995, and this usually means that we will be needing even more client resources to accomodate these (see HARDWARE CONFIGS). New installations will be using the new 700 series hardware and also utilize a local swap disc to provide resources for more and larger applications. We would like to provide a new revision of the GUI and toolset every six months to keep abreast of the latest available technology. Lastly, we plan to replace 50 % of the 1400 remaining OLTP terminals with workstations in the next year.

A method to address many significant concerns has been labeled "COE", for common operating environment. This implies a "standardization" of the platform and software that can provide flexibility to the customer while significantly enhancing supportability. A project to define a COE for desktop workstations in currently underway.

TECHNICAL ASPECTS

The following section describes some of the more technical aspects of integrating workstations into the existing emvironment. In general, a workgroup was placed on the same cluster server so they could share a common set of configuration files. See figure 2 for a graphic illustration of a cluster and [3] for more information about clusters.

Security is always a significant concern, and workstations were the cause of even greater than normal attention, due to recent history like the Internet worm and other breaches of UNIX security. We used three major strategies to ensure adequate security and yet keep users from having security get in their way. The first was to allow MPE passwords to be contained in files on the system, but keep direct access to these files limited to the VUE

administrator. This provided the functionality that was needed along with strict control over who could see those passwords. See figure 5 for how this is done. The second was strict control over access to the cluster servers, and consequestly the cluster clients also. Access is limited to the minimum required, even to the point of having these machines perceived as "unfriendly" by other network users. This also includes the fact that in general, we did not give users access to HP-UX at all, except through the VUE environment. Last, we charged the users with being responsible for "locking" their workstations' display when it was not being used, and also providing physical security in general.

HARDWARE CONFIGURATION

The server in a diskless cluster configuration will perform the following tasks:

- startup the diskless clients
- manage the file system for the entire cluster
- allocate and manage swap space (assumes no local swap on the clients)
- provide a backup mechanism to ensure effective system recovery

The client in a diskless cluster performs the following tasks:

- locally runs the operating system (HP-UX)
- manages all processes locally
- provides the X-windows seat and interface for the end user

All these workstations are configured into what Hewlett-Packard calls a cluster (see [3] for more information). This means that, for a workgroup, there is one server that boots all the workstations and provides a shared file system that all the machines can use. This topology provides a simpler system to administer since there is only onefile system, while still allowing a significant amount of autonomy between clients (or "cnodes", aka the machine that sits on the user's desk).

The cnodes, also known as discless clients , consist of a unit with processor, memory, I/O etc., a

bitmapped graphics display, a keyboard and a mouse. The server usually had just the SPU, memory and I/O along with the disc(s), but no bitmapped graphics display ... we used a terminal as the system console to lower the cost. For more demanding applications and for sites where network traffic was already high, a local swap disc can be added to each cnode to improve performance (see Figure 1).

To enhance reliability at those sites with more than a single cluster, the cnodes for a workgroup are configured to have every other cnode boot from another server (see Figure 2). In this way, if a server crashed,only half of the workgroup would be out of service. This required "cross-mounting" of the file systems between servers and the use of Hewlett-Packard's VHE (virtual home environment) to implement (see [5]), but the extra measure of reliability was well worth the effort. In addition, this provided an environment where a user could sit down at any workstation in the group, use their standard login, and get his or her proper environment.

The numerical relationship of clients to server can not be stated as an exact recommendation. Many factors impact the number of clients that can be handled by one server such as: display buffer size when usingterminal emulation, the number and size of the windows, and the amount of swap space on the server. More swap space is needed to support more clients. While no hard recommendations can be made concerning the exactnumbers of clients supported per server, a typical range, offering good performance, seems to be 20 to 30. This is based on the community experiences of the WBP taskforce. Your mileage may vary.NAFO is presently using ratios of 20 workstations per cluster server for 400 series machines and 40 per server for 700 series machines.

SOFTWARE CONFIGURATION

For those workgroups with a small number of people, provisions were made to allow several workgroups to share a single server. This was accomplished by using a flexible file system structure based on "levels". The levels were global (for all of NAFO), site specific, group specific and individual. In this way, we could provide almost any combination of

software to a user and still have a system that was manageable (see Figure 4).

As mentioned previously, the VUE administrator was responsible for any changes to the environment for the workgroup. This was accomplished by using a single set of configuration files residing in a "group" directory that all users in that group would use by default. However, use of these group configuration files could be overridden simply bylocating an individual configuration file in the users home directory. Likewise, if no group configuration file existed for a certain application, the system was designed to look "up" the file system directory tree and get configurations from the sitewide or NAFO directory. This method provided maximum flexibility for the user while remaining relatively simple to administer.

To aid in making changes to VUE, a tool was created that allowed the VUE administrator to automate some of the more complex components. For example, to change the text color for a certain window, the VUE administrator may need to edit the file called: "/usr/local/atc/users/config/app-defaults/HPterm". This includes the path name of course, but they would need to remember this. In addition, any changes made may need to be copied to other servers if the group inquestion was large. Instead of requiring that all VUE administrators actually type this hideous string of characters and also make sure any changes were propagated to other servers if needed, the tool was written to recognize the particular file system structure used and make use of it. This resulted in the user simply choosing from a menu item like "Change a resource file", and then picking the one that was "HPterm". It is not difficult to see that this made the VUE administrators life much easier. However, it took some amount of expertise to craft such a tool.

AUTOMATED LOGINS

To provide the "touch of a button" logins that saved so much time while still providing individual information to MPE, we used a locally written program called winx. Winx was essentially a "front-end" for launching a window that automatically logged on to a remote machine. Upon invocation, winx looks

first in the users home directory for the user-specific login information (phone extension, initials, etc. in .dotinfo), and then searched for the auto-login file that contained the actual logon string (a HELLO user.account for MPE) in (1) the users HOME directory, (2) the group directory, (3) the site directory, and finally (4) the NAFO directory. Once the logon string has been found, winx combines the user-specific information and the logon string to form an individual logon string in the correct format, then process handles a new window onto the users display that connects to the proper machine and provides the proper logon (refer to Figure 5).

To make things as easy as possible for the VUE administrator, this entire process can be condensed into the following definition for a button that access a remote application :

```
applic1  [P]  @app1.bm  f.exec "win3x -h hostname  -a
applic1"
```

In order from left to right, this definition specifies the button name as "applic1" , the [P] specifies that this is a pushbutton (see [2] for more info.), and the "@app1.bm" defines the bitmap with which to label the pushbutton. The f.exec syntax simply means "execute the following command as if I had typed it on the command line". Winx only needs the two parameters that define which host to connect to and which auto-login file to use, the individual information is gleaned from the user ID that invoked the winx program, and which kind of remote machine is taken from the name used to invoke winx (in this case win3x, meaning connect to an HP3000).

You can see that using winx not only provided significant functionality and flexibility in automating logons, but also minimized the effort and training required to administer automated logins (see Figure 5 for a picture of how this works).

THE VUE FRONTPANEL

For those familiar with Hewlett-Packard VUE (visual user environment), you will see from the example in Figure 6 that we have customized the frontpanel significantly. For those who haven't seen VUE before,

it was designed to be "intuitive" in its use, that is, the user simply clicks the mouse while the mouse cursor points at the item they wish to activate, simulating the physical press of a button. If your end users already know how to use a mouse, training them to use VUE is relatively simple, if not, this is the first skill that they must learn in order to use the VUE environment successfully.

Please note that all the examples and discussion here refers to version 2.0 of VUE. The current version is 3.0 which not only has significant additional functionality over 2.0, but also uses a different syntax for the configuration file. It is for this reason that we have not yet moved to the 3.0 version, as this will require changing some utilities and re-training the VUE administrators so that they will work properly with and understand the new syntax. NAFO is presently still working on this conversion.

The top row in this example (Figure 6)consists of some typical "utility" functions that come with Hewlett-Packard VUE such as a clock, a calendar, calculator, SharedX, and so on. There is also a "lock" button (bottom right hand corner of Figure 6) that allows a user to lock the display while they are away from the desk. We also provided a caps lock indicator on the frontpanel since this was absent from the standard set of VUE indicators. This is shown on the top row.

You will also note that in the middle of the top row there are four buttons labeled "One" thru "Six". These are for accessing multiple "workspaces", which is basically a way to provide extra screen real estate for those who may have so many windows active on the screen that it gets cluttered. This functionality is only provided by Hewlett-Packard VUE and is not expicitly a part of the X window system. In practice, however, we found that most users did not take advantage of the extra workspaces, except for the GEM team who needed access to sales, service, contracts, and product information and had a workspace for each.

So, we have seen how each button corresponded to a line in a confguration file like the fragment shown here (this one was used for the example

frontpanel show in Figure 6 - it defines only the bottom row) :

```
Row Bottom
{
 Logo    [P] @logo.m.bm =115x30 f.version
 Desk    [P] @desk.bm =78 f.exec "win3x -h ise71 -a
             hpdesk"
 Cris    [P] @cris.bm f.exec "win3x -h ise27 -a cris"
 Pal     [P] @pal.bm f.exec "win3x -h ise29 -a pal"
 Iqs     [P] @iqs.bm f.exec "win3x -h ise33 -a iqs"
 Oms     [P] @oms.bm f.exec "win3x -h ise58 -a oms"
 Sports  [P] @sports.bm f.exec "win3x -h ise43 -a
             sports"
 Armada  [P] @armada.bm f.exec "win3x -h ise45 -a
             armada"
 Patsy   [P] @patsy.bm f.exec "win3x -h ise17 -a patsy"
 TrakII  [P] @trakii.bm f.exec "win3x -h ise78 -a
             trakii"
 Lock    [P T] @lock.m.bm  f.action LOCK_DISPLAY
 vuewmbusy [P R] @exit.m.bm @exit2.bm =59 =13x8+25+12
                 f.action  EXIT_SESSION
}
```

In general, we provided a frontpanel button for each application accessed by a particular workgroup at the initial install. Once the group was up and running, the VUE administrator had control over how the frontpanel looked at what items or buttons appeared in it. This "local control" of the frontpanel provided two benefits, the process owners had control over their environment, and IT resources were not required to make moves, adds and changes on a day to day basis.

For those groups that required additional functionality or had a very large number of applications, we could either add another row of buttons to their frontpanel, or have buttons correspond to a group of applications rather than a single one. See [2] for more detail on how this can be accomplished using the VUE file manager.

THE RESOURCE FILES

In this framework of VUE, the X window system, and the HP-UX operating system underneath it all (see Figure 7), one way to control an individual

applications appearance (text color, background color, window size, etc.) is through the use of resource files (again, see [1]). These are just ASCII files that define how a specific applicationshould appear. For example, the listing below shows an example resource file for hpterm. Hpterm is the standard Hewlett-Packard terminal emulator on the workstation, and we might want it to look a certain way in order to facilitate using a specific HP3000 application. In this example, we use "geometry" to indicate that we want an 80 column by 24 line display window. This can be very important when using VPLUS applications since any other window geometry will not provide what the program expects to be there. Likewise, using the "cursorColor"resource, we define the text cursor to be red in color. This came about after several people complained that they lost track of the cursor when using applications that used both inverse video and half-bright fields.

Also of interest is the "TermID" resource. This controls the reply to a terminal ID status request from the HP3000. The default is to tell the HP3000 that it is a 2622 terminal, which did not always achieve the desired results. Hpterm provides for two other possibilities : a 2392 and a 700/92. We found that most of our applications worked best with this resource set to 70092, but a small minority worked best with the 2392. Note that the resource name is correctly specified as "70092", even though the terminal designation is "700/92". Again, the idea is that the local VUE administrator can modify these resources to maximize the usability of the environment for his or her workgroup.

```
HPterm*foreground:              green
HPterm*background:              black
HPterm*cursorColor:             red
HPterm*font:                    hp8.7x10
HPterm*geometry:                =80x24
HPterm*saveLines:               5s
HPterm*scrollBar:               True
HPterm*scrollBar*background:    black
HPterm*scrollBar*foreground:    LightGray
HPterm*softkey*background:      gray
HPterm*softkey*foreground:      black
HPterm*termId:                  70092
HPterm*title*font:              user11x19
```

Sample App-defaults file for hpterm

There is also a resource file for the window manager, VUE. I mentioned in a previous section how we had to customize this to keep users from resizing windows that should always be 80 by 24. This was accomplishedby making three changes to the standard VUE configuration. First, the window border resize handles and the maximize button were disabled, toprevent the mouse from resizing a window. This is shown in A below. Second, the keystrokes, also known as keyboard accelerators, to do this were disabled also. This is shown in B below. Last, the window menu items that provided this function had to be removed also. This was done using the vuewmrc file (see [2]) and is not shown.

A] Vuewm*HPterm*clientDecoration: -maximize -resizeh

B] Vuewm*HPterm*clientFunctions: -maximize -resizeh

REMOTE SYSTEM MANAGEMENT

Disc space and performance -

Today the tools in use for space management are 1) a locally developed shell script 2) HP-UX supplied disk quotas. The script runs every evening and mails the System Management team messages on those machines that have exceeded 90% of disk capacity. On all discless file and LM/X servers disk quotas have been enabled to restrict unwanted disk usage. Any user that exceeds their quota is notified automatically at login by the appearance of a window informing them of their present disc usage statistics.

Performance monitoring is accomplished with Laser/RX and Glance. Laser/RX is utilized to profile a machine over a specified period, usually a week in duration. This tool requires a PC that has the analysis tool installed. There are two such PCs in the System Management group. The data produced can also be filtered through SAS, a statisticalsoftware package, for further evaluation. The SAS based analysis is performed by the consultants on the operations staff.

Repair of remote systems -

All remote servers are under a 4 hour response SSO support contract. The systems management group works with both the local site contact and the CE assigned the hardware call to resolve the problem.

Backups -

The tool selected by the Systems Management group is the fbackup and frecover utilities provided with HP-UX. On all remote servers cron is utilized to launch the backup. The script mails errors encountered to the systems management group for problem resolution. This may be as simple as having the OA group inform their contacts that tapes were not inserted in the DAT DDS tape drives.

HP3000 IMPACT

The ability to create multiple windows and switch between them does not come without some cost on the HP3000. According to the division there are two resources that are consumed by the additional sessions that result from the multi-window environment. Excessive reduction in either of these two resources can result in severe performance problems on the HP3000 that can more than offset any productivity gains made by the implementation of workstations.

VT (NS) Sessions

For MPE-XL release 3.0, the maximum number of VT (NS) Sessions allowed is 600. At a minimum, 3 processes per VT session exist: JSMAIN, CI, and VTServer. Normally we will use more than just the CI, so an additional process is utilize d for the application program. This would bring the total to 4 processes per VT session. If the maximum number of VT sessions and the maximum number of system sessions were attempted, the process limit of 3119 total system processes could be exceeded. That is 600 VT sessions times 4 processes plus 250 DTC sessions times 3 processes for a total of 3150 processes. (The 3 processes for the DTC session are JSMAIN, CI, and a user process. Approximately 100 processes are normally reserved for system usage).

A remote logon to another XL system, (logon via a virtual terminal), requires 3 processes on the host

system and 3 on the remote system. These are a JSMAIN, a CI, and a VTSERVER process on both sides. If the user runs a simple program (which does no process handling), then there would be a total of 4 processes per logon.

As stated above MPE-XL Release 3.0 supports 600 VT sessions. With 600 active VT logons, it is typical that 2400 processes would be used. With a maximum of 3119 processes it is NOT quite possible to achieve 850 active logons if 600 of these active logons are VT logons: 3119 - 100 system processes - (600 * 4) - (250 * 3) = -131 processes. Therefore with 600 active VT logons it is estimated that a maximum of 206 additional active DTC logons are possible.

The point here is don't assume you will be able to achieve the system limits of 850 sessions with a significant number of workstation users. You will quite likely run up against the limit of 3119 system processes first. You must also consider the CPU usage (see below) when calculating the number of users and impact on the system.

Below is a worksheet which can be used to estimate the concurrent process requirements for an example system. If your active users are running programs which do process handling, then YOU WILL NEED TO CHANGE THE MULTIPLIERS used for active jobs and sessions (local or remote).

# Active Jobs	_____	(<= 850) x 3 =	_____
# Inactive Local Sessions	_____	(<= 850) x 2 =	_____
# Active Local Sessions	_____	(<= 850) x 3 =	_____
# Inactive Remote Sessions	_____	(<= 600) x 3 =	_____
# Active Remote Sessions	_____	(<= 600) x 4 =	_____
	+	system processes	
	+ 100		
Total Connections	_____	(<= 850) Total Processes	_____ (<=3119)

It is very important to note that many applications and third party products use process handling. YOU MUST DETERMINE HOW MANY ACTUAL PROCESSES WILL BE USED BY EACH SESSION OR JOB IN

ORDER TO ACCURATELY PREDICT THE NUMBER OF USERS THAT THE SYSTEM WILL SUPPORT. The numbers listed above are only used to illustrate the issue and assist you in the thought process necessary to determine the actual limitations you may be facing. Don't assume that your application or system will conform to the above example.

Additional CPU time

Virtual terminals from the workstation use about 1.5 times more CPU per session for the I/O portion of the applications activity. If the machine you want to access via workstations is a heavily loaded classic, don't do it! Adding workstations to such a situation will only make a bad situation much worse.

Since only the I/O portion of the application consumes such resources, the impact of adding workstations varies dramatically from application to application.

An application that uses one or two screens for data entry will not see much of a change in CPU utilization. However, an application that the users constantly toggle from screen to screen could see a significant impact on the HP3000's performance.

System Managers should run Laser-RX on their HP3000 before workstation implementation. This will help determine the current overhead created by DTCs on their machines. They should then do some simple modeling to see what impact the additional virtual terminals may have on their specific application mix.

It can't stressed enough that this planning be done correctly up front prior to any workstation purchase decision is made. If you are not sure how to do this, get help either from your local IT staff or subcontract out to Hewlett-Packard's PSO.

A System Resource Payback Related to Workstations

Logging on and off an HP3000 uses a significant amount of system resources for a short duration. If the pre-workstation environment has users repeatedly logging on and off of machines to use various

applications throughout the day, the single log on aspect of using workstations can result in a savings of system resources. It is very difficult to determine to what degree this savings will offset the additional resource needs detailed above. The effect will be greater at sites with greater numbers of users performing greater numbers of repetitive logons.

SUMMARY

Well, I hope that this article has been informative and not too boring. I tried to cover everything at least once, while emphasizing those ideas, decisions, and strategies that were helpful in insuring success in this case study. To reiterate, the critical elements of success were:

- Team effort and committment. Nothing can take the place of dedicated and knowledgeable personnel. A balance of technical and process knowledge is necessary to yield the best results.

- Try new methods on a small scale first. The pilot projects were invaluable in not only demonstrating what worked, but in pointing out what the pitfalls were as well.

- Standardize. Implementing new technology is much easier if one can leverage off the efforts of others and save "reinventing the wheel" every time.

REFERENCES

Many thanks to everyone at all the pilot sites, the Client Computing Group, Corporate IT, the Site Networking Group, especially the members of the WBP Task Force, and all of those who suffered through one of my 2-day "train the trainer classes".

[1] "Using the X Window System", Hewlett-Packard p/n B1171-90037, edition 4, February 1991

[2] "HP Visual User Environment System Administration Manual", Hewlett-Packard p/n B1171-90023, edition 1, February 1991

[3] "Managing Clusters of HP9000 Computers Sharing the HP-UX File System", Hewlett-Packard p/n B1862-90006, First Edition, January 1991

[4] "Using Network Services", Hewlett-Packard p/n B1012-90009, February 1991

[5] "Installing and Administering NFS Services", Hewlett-Packard p/n B1013-90001, September 1989

[6] "Workstation Best Practices", Mike Holland, Hewlett-Packard internal document, September 1991

[7] "HP SharedX User's Guide", Hewlett-Packard p/n B2305-90536, first edition, January 1992

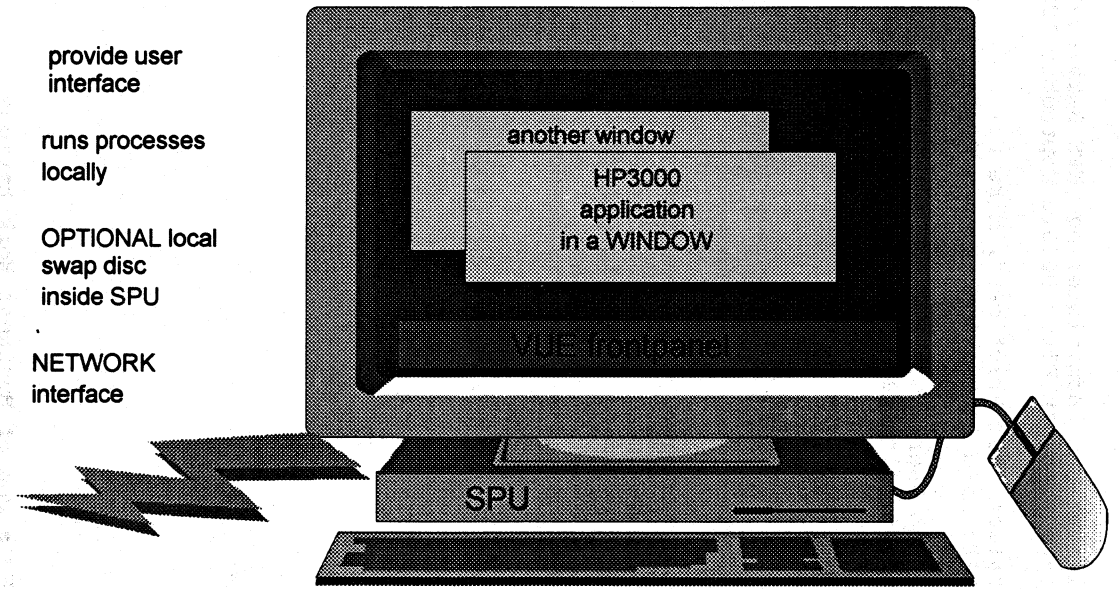


Figure 1.
Workstation hardware
HP's Use of Business Workstations
7008-28

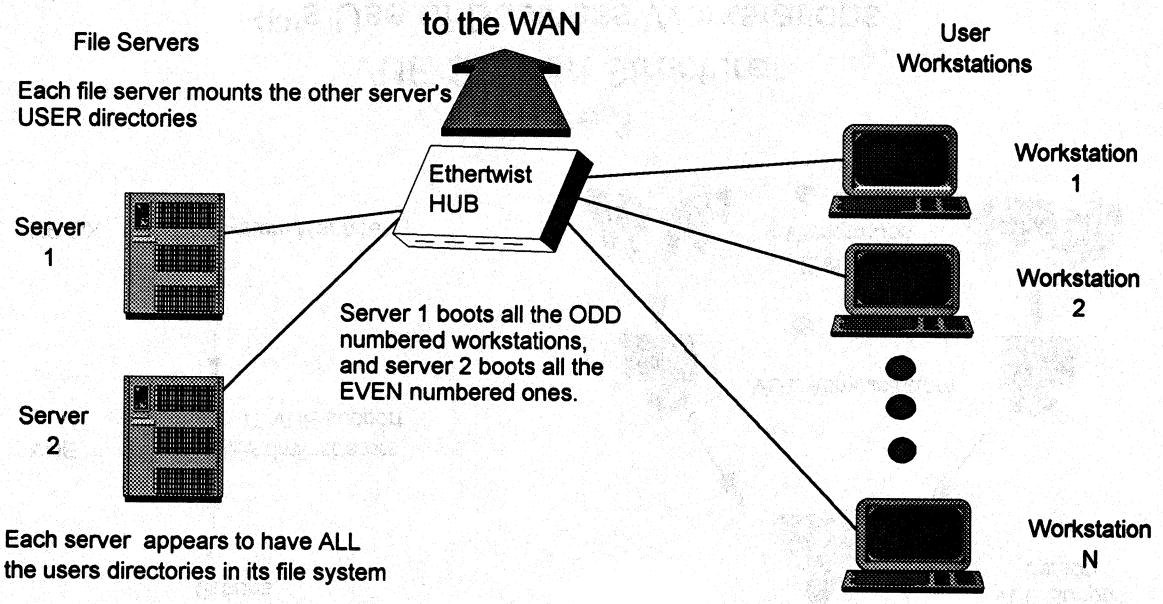


Figure 2
Remote Office Topology
HP's Use of Business Workstations
7008-29

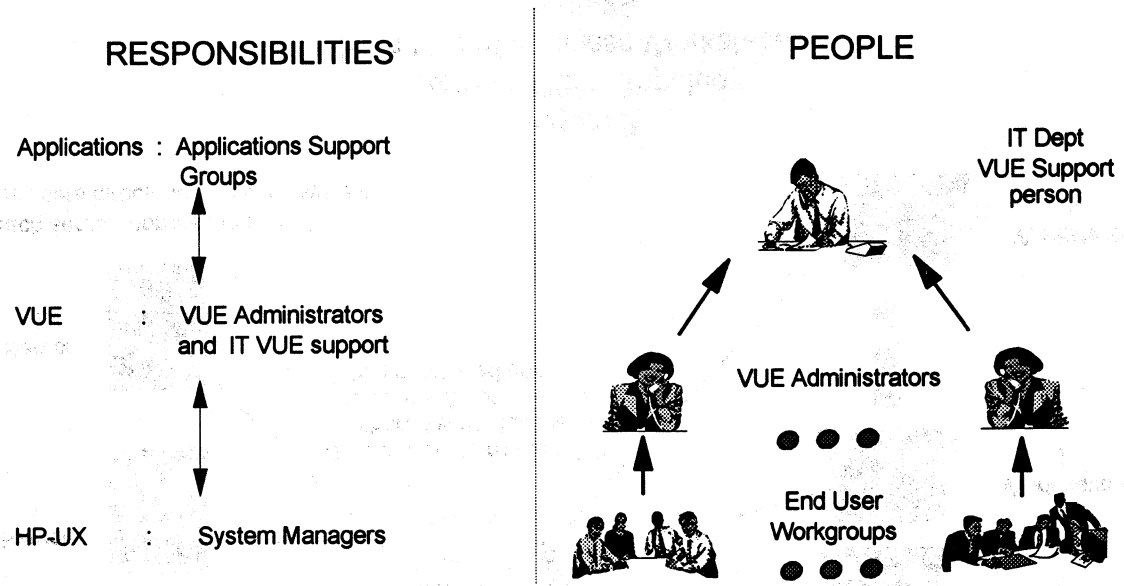


Figure 3
VUE Support Structure
HP's Use of Business Workstations
7008 - 30

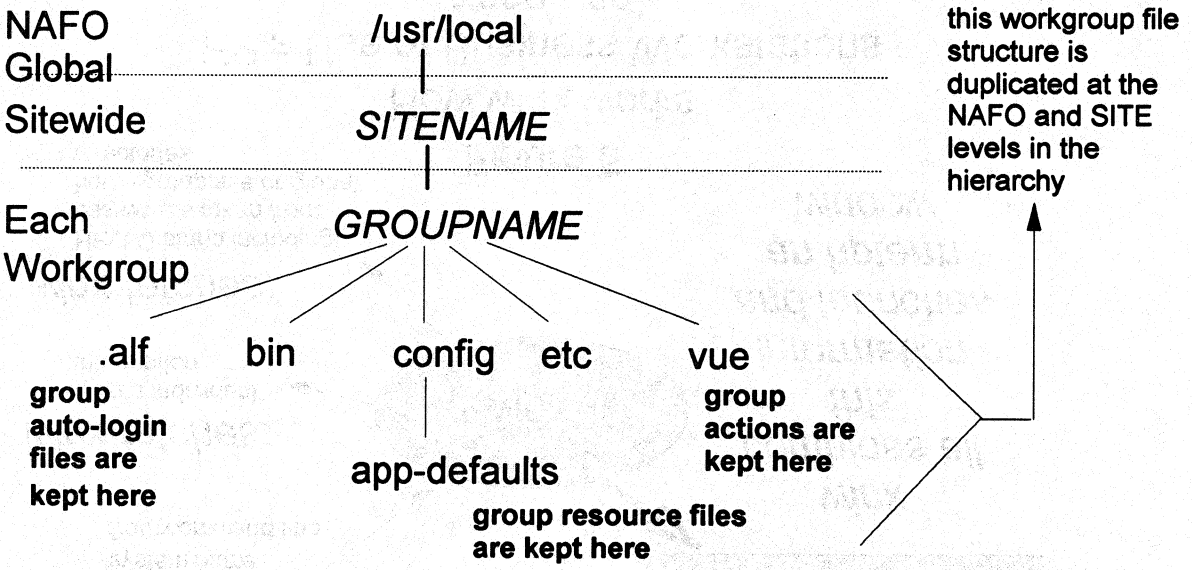


Figure 4

NAFO File System Structure

HP's Use of Business Workstations

7008 - 31

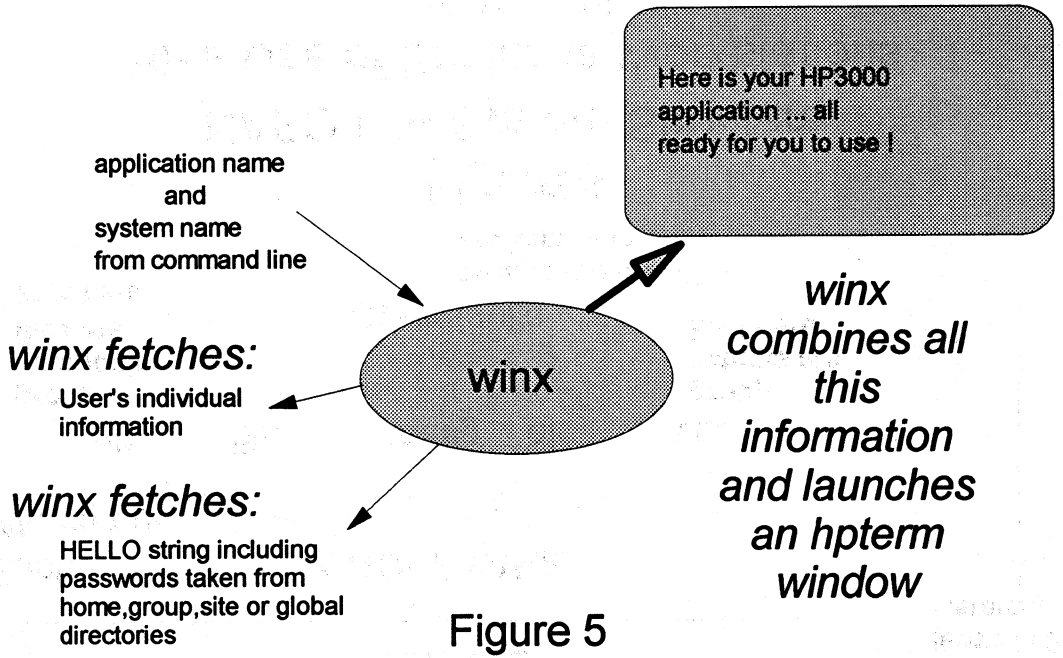


Figure 5
How winx works
HP's Use of Business Workstations
7008 - 32

END USERS

|

VUE Window Manager 2.01

|

X Window System - X11R4

|

Operating System - HP-UX 8.0

|

Physical Hardware - Series 400/700

Figure 7 System Organization

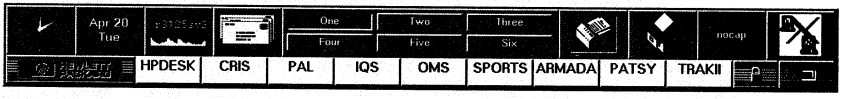


Figure 6 Example VUE Frontpanel

GUIDELINES FOR WRITING DCE APPLICATIONS IN COBOL

Paper #7009
Charles Knouse
Information Networks Division
Hewlett-Packard Company
19420 Homestead Road Cupertino, CA 95014
(408) 447-2354

INTRODUCTION

The Distributed Computing Environment (DCE) from the Open Software Foundation (OSF) is emerging as the dominant framework for open distributed client-server applications. DCE will soon be available on a wide variety of platforms, including:

HP3000/900 systems (MPE/iX)
HP9000/700 workstations and HP9000/800 systems (HP-UX)
IBM workstations (AIX), PCs (OS/2), and mainframes (MVS)
DEC workstations and servers (OSF/1)
Sun workstations and servers (Solaris)[1]
Many other Unix-based systems (Bull, SNI, USL, Pyramid, etc.)
Intel-based personal computers (MS-DOS, Windows, NT, SCO Unix)[2]

Developers of business applications may want to use DCE when writing new or upgrading existing applications. However, the most popular business programming language, COBOL, is not directly supported by DCE. The DCE services and tools are designed to be used with the C programming language. This paper will provide some simple guidelines for developing DCE applications using COBOL. Included will be COBOL-callable templates for using DCE services and rules for writing Interface Definition Language (IDL) descriptions for calling remote procedures in COBOL. An example COBOL application following these guidelines will be discussed.

The discussions here will be limited to existing DCE tools and interfaces. These guidelines can be used today with the DCE 1.0 software. I will touch upon some possible enhancements to DCE for easier use by COBOL at the end of the paper.

---

1. DCE on Sun is supplied by Transarc Corporation.
2. DCE client-only software for MS-DOS PCs is available from Gradient Technologies. Also, the Microsoft Remote Procedure Call (RPC) facility is modeled after and can inter-operate with DCE RPC.

EXAMPLE APPLICATION

I will use a simple example application to illustrate the guidelines discussed here. This application provides order entry clerks on-line access to data on stock parts in a warehouse and allows the clerks to ship parts to fill customer orders. In its original form (shown in Fig. 1), the application is a monolithic program running on one system. It accesses two databases: the stock database (part number, description, and price of each stock part) and the warehouse database (part number and quantity on hande of each part in the warehouse). The application provides a terminal screen interface for each clerk.

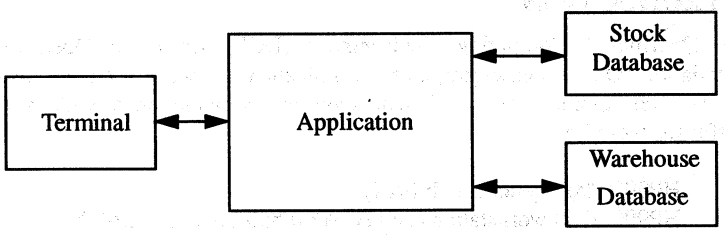


Figure 1. Original Example Application.

The COBOL program for the original application is shown in File 1. (ORIGCOB)[3]. Only the top level of the program is shown. The internals of the major paragraphs are omitted. The data declaration file RECTCOB will be discussed later.

```
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
$INCLUDE RECTCOB
01 EXIT PIC X(3).
. . .
PROCEDURE DIVISION.
BEGIN.
    PERFORM OPEN-STOCK-DB.
    PERFORM OPEN-WAREHOUSE-DB.
    PERFORM OPEN-TERMINAL.
```

3. Files for this example include both MPE/iX files (record size 80 bytes; 8 char max name) and POSIX files (byte stream, long name, etc.). The distinguishing convention is MPE/iX file names are upper case and POSIX file names are lower case.

```
    PERFORM PROCESS-TRANSACTION UNTIL EXIT = "YES".
    PERFORM CLOSE-TERMINAL.
    PERFORM CLOSE-STOCK-DB.
    PERFORM CLOSE-WAREHOUSE-DB.
    STOP RUN.

. . .
PROCESS-TRANSACTION.
    PERFORM INPUT-TRANSACTION.
    PERFORM READ-STOCK-RECORD.
    PERFORM READ-WAREHOUSE-RECORD.
    PERFORM SHIP-AND-BILL.
. . .
```

File 1. ORIGCOB: Original COBOL Application.

Now suppose we want to distribute this application over several computers. Suppose there are now a number of warehouses, each with its own database and computer, and there is a separate computer to hold the stock database. We could use a distributed database with the original monolithic program. Or we could split the application into client and server programs. The pros and cons of each of these approachs is beyond the scope of this paper. For the purposes of illustration I will chose the client-server approach, using DCE to tie the distributed application together.

The desired client-server configuration is shown in Figure 2. For each type of database, there is a server that accesses the database: one STOCKDBD program for the Stock database and several WHOUSED programs for each Warehouse database[4]. I could in fact have several STOCKDBD programs tending replicated Stock databases for high availability, but for simplicity I will ignore this possibility. The CLIENT program provides the user interface for the order clerk. This initially will be the terminal screen interface from the original application, but in the future it could be upgraded to a Windows interface on a PC or a Motif interface on a workstation. The CLIENT retrieves data from the STOCKDBD and WHOUSED servers using the DCE Remote Procedure Call (RPC) facility.

In this as in any client-server application, there must be a way for the servers to advertise the services they provide and for the clients to find the appropriate services and servers. DCE provides the *RPC Name Space* and the underlying *Cell Directory Service (CDS)* for this purpose. The STOCKDBD and WHOUSED servers

4. The D suffix in the program names is taken from the Unix world, where it indicates a program is a daemon, which usually runs in the background to service user requests. Note that the programs are MPE files. Currently MPE/iX restricts programs to MPE (not POSIX) files.

can register their locations, and the CLIENT can find them, using RPC name space calls.

DCE also provides security facilities that can be used by the example application. The servers and users can be assigned principal identities that are protected by passwords known only to the server or user and the DCE Security Server SECD. Servers and users then login to DCE to establish and verify their identities. The application can use *Access Control Lists (ACLs)* to control which users can access which servers and databases.

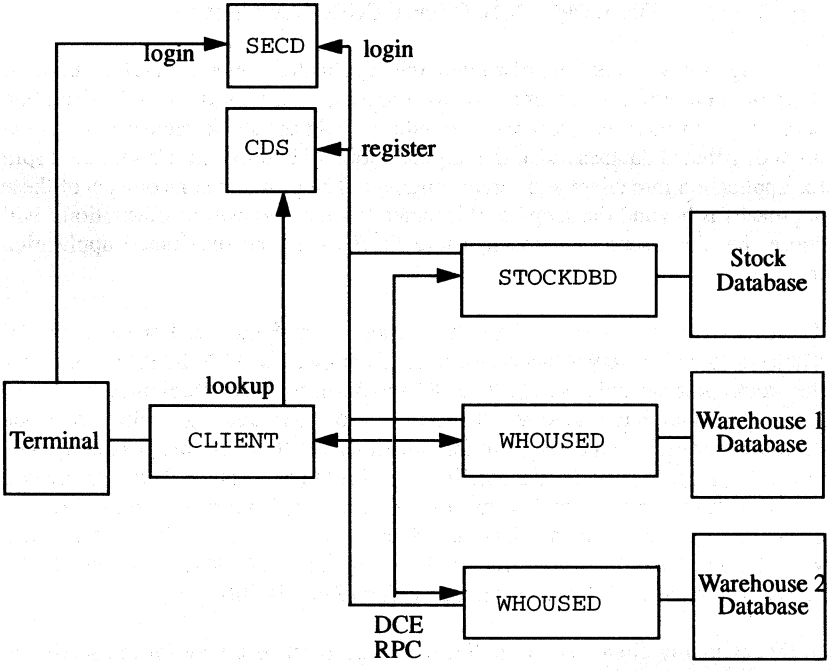


Figure 2. Client-Server Example Application Using DCE.

Now I will discuss in more detail the steps to turn the original application into the client and server programs. The two major categories are Interface Design and Client-Server Design.

INTERFACE DESIGN

An *interface* is, in general terms, how the clients and servers communicate -- how the client requests a service, and how the server performs the service. More specifically, in DCE RPC an interface defines a collection of remote procedures that a server can execute on behalf of its clients. An interface is defined using the *Interface Definition Language (IDL).* IDL is a declarative language closely modeled after C, with extensions for things needed by RPC. An IDL file contains a header with general interface attributes, data type definitions, and declarations for each remote procedure in the interface. Each procedure declaration supplies a name, optional procedure attributes, and a list of zero or more parameters. Each parameter has a data type, parameter attributes, and a name. Among the most important parameter attributes are `in` and `out`, which specify the direction of the parameter transmission. `in` parameters are sent from the client to the server; they are the data needed by the server to perform an operation. `out` parameters are sent from the server back to the client; they are the results of the operation. A parameter can be both `in` and `out`.

The first step in interface design is to determine what remote procedures are required and what their parameters should be. With procedure-based languages like C, this is normally straight-forward. The procedures may already exist. But with traditional COBOL this step may be more complicated. Hopefully, the COBOL program is well-structured in paragraphs that can become the basis for the remote procedures. In our original application, the paragraphs `READ-STOCK-RECORD`, `READ-WAREHOUSE-RECORD`, and `SHIP-AND-BILL` look promising.

Traditional COBOL programs tend to declare data items globally in the `DATA DIVISION` that are shared among the `PROCEDURE DIVISION` paragraphs. But this violates a cardinal rule of RPC interface design: no shared memory between clients and servers. All data must be explicitly passed as `in` or `out` parameters. So we must carefully determine which data items are used in the target paragraphs and include them as parameters in the remote procedure declarations.

Once the interface procedures and their parameters have been identified, we can write an IDL file for the interface. But now we face a major obstacle. IDL is so like C that we have to translate COBOL data definitions into C data types. But COBOL provides many data types and formats that cannot be cleanly translated into C. My advice is to stick to a small subset of data types that can be translated. Table 1 shows some elementary data types.[5]

---

5. Based on Gordon, Shawn M., *C from a COBOL Perspective*, Part 1, HP Chronicle, March 1993.

Table 1: COBOL to IDL Data Type Translation.

COBOL	IDL	Size (bytes)
`PIC S9(4) COMP`	`short int`	2
`PIC S9(9) COMP`	`long int`	4
`PIC X(`*n*`)`	`char [`*n*`]`	*n*
`PIC S9(`*n*`) COMP-3`	`byte [`*n*`]`[a]	*n*
not available	`float`	4

a. COMP-3 data-items are packed decimals, which are not directly available in IDL. They can be transmitted as opaque byte strings using the `byte` data type. This guarantees that the data bytes will not be converted during parameter transmission.

A major COBOL feature is the *record*, which is a sequence (possibly nested) of data items of arbitrary types. Corresponding to the COBOL record is the IDL `struct`. Each record field becomes a `struct` field. Files 2 and 3 shown the COBOL records used by our application and the corresponding IDL data declarations. The IDL `typedef` statements define the simple data types used for record fields and the structs matching the records. Note well the `FILLER` fields added to the records and structs to ensure that the `long int` values are "naturally" aligned on four byte boundaries. This will turn out to be required for the correct transmission of these records by RPC.

```
01 STOCK-RECORD.
   02 STOCK-NUMBER          PIC X(10).
   02 STOCK-DESC            PIC X(40).
   02 FILLER                PIC X(2).
   02 STOCK-PRICE           PIC S9(9) COMP.
01 WAREHOUSE-RECORD.
   02 STOCK-NUMBER          PIC X(10).
   02 FILLER                PIC X(2).
   02 QTY-ON-HAND           PIC S9(9) COMP.
01 TRANSACTION-RECORD.
   02 CUST-NAME             PIC X(40).
   02 CUST-ADDRESS          PIC X(100).
   02 STOCK-NUMBER          PIC X(10).
   02 FILLER                PIC X(2).
   02 QTY-TO-SHIP           PIC S9(9) COMP.
```

File 2. `RECTCOB`: COBOL Record Definitions.

```
interface record_types
{
    typedef char stock_number_t[10];
    typedef char stock_desc_t[40];
    typedef long int price_t;
    typedef long int qty_t;
    typedef char name_t[40];
    typedef char address_t[100];
    typedef char filler2_t[2];

    typedef struct {
        stock_number_t   stock_number;
        stock_desc_t     stock_desc;
        filler2_t        filler;
        price_t          stock_price;
    } stock_record_t;

    typedef struct {
        stock_number_t   stock_number;
        filler2_t        filler;
        qty_t            qty_on_hand;
    } warehouse_record_t;

    typedef struct {
        name_t           cust_name;
        address_t        cust_address;
        stock_number_t   stock_number;
        filler2_t        filler;
        qty_t            qty_to_ship;
    } transaction_record_t;
}
```

File 3. `record_types.idl`: IDL Record Definitions.

Now that I have set up IDL declarations for the relevant data, I can write the IDL files for the interfaces. Since I have two servers, I will chose to have two interfaces: `stock_db` and `warehouse`. Table 2 shows the procedures in each of the interfaces and the corresponding paragraphs from the original application.

Table 2: Remote Procedures and Original Paragraphs

Interface	Remote Procedure	Original Paragraph
`stock_db`	`read_stock_record`	`READ-STOCK-RECORD`
`warehouse`	`read_warehouse_record`	`READ-WAREHOUSE-RECORD`
`warehouse`	`ship_and_bill`	`SHIP-AND-BILL`

Note the transliteration of the COBOL paragraph names into C-style procedure names: the - becomes _ and uppercase becomes lowercase. (This same style of transliteration was applied to the data item names.)

Files 4 and 5 show the completed IDL files for the interfaces. Each has an interface header the specifies the uuid (*universal unique identifier*) and version the RPC uses to uniquely identify the interface. Each uuid is an encoding of the location where and the time when the interface was created. The uuid can (and should) be generated using the `uuidgen` program. In fact, the POSIX shell command `uuidgen -i > X.idl` will generate the start of an IDL file for the interface `X`. Each IDL file also includes `import "record_types.idl"`, which brings in the record definitions needed for the interface.

```
[uuid(7E3F8A8E-5538-11CC-A91A-080009226619,
 version(1.0)]
interface stock_db
{
    import "record_types.idl";

    void read_stock_record(
         [in] handle_t *h,
         [in] stock_number_t stock_number,
         [out] stock_record_t *stock_record,
         [out] error_status_t *st);
}
```

File 4. `stock_db.idl`: Stock_db interface definition.

```
[uuid(83C9B7CC-5538-11CC-8C00-090009226619),
 version(1.0)]
interface warehouse
{
    import "record_types.idl";

    void read_warehouse_record(
         [in] handle_t *h,
         [in] stock_number_t stock_number,
         [out] warehouse_record_t *warehouse_record,
         [out] error_status_t *st);

    void ship_and_bill(
         [in] handle_t *h,
         [in] stock_record_t *stock_record,
         [in] transaction_record_t *trans_record,
         [out] error_status_t *st);
}
```

File 5. `warehouse.idl`: Warehouse interface definition.

Let's examine the `read_stock_record` procedure in `stock_db.idl`. The `void` before the procedure means that it will have no return value. The first parameter, `h`, is the *RPC binding handle*, used by RPC to locate and keep track of the server. I will discuss binding in more detail later. Note well that the address of the handle is passed (`*h`). This is required by the COBOL calling convention, and is different from the normal convention for passing the handle. The second parameter, `stock_number`, is an input parameter that specifies what part to find in the database. Its type `stock_number_t` corresponds to the COBOL type `PIC X(10)`. The third parameter, `stock_record`, is an output parameter to receive the data for the stock part. Its type `stock_record_t` is a structure with fields `stock_number`, `stock_desc`, and `stock_price`. Since a record will be passed back through this parameter, the address of the parameter (`*stock_record`) must be specified. The fourth and last parameter, `st`, is of the RPC-defined type `error_status_t`. It is an output parameter to be used to return the status of the remote procedure call. I will say more about this later.

Connected with the error status parameter `st` are the parameter attributes `fault_status` and `comm_status`. These indicate to RPC that an error is to be returned to `st` if there is a fault or a communications failure during the remote procedure call. These attributes are not included in the IDL files, however. They are specified in separate files called *Attribute Configuration Files (ACFs)*. The ACFs for our two interfaces are shown in Files 6 and 7.

```
interface stock_db
{
read_stock_record([comm_status,fault_status] st);
}
```

File 6. `stock_db.acf`: Stock_db interface additional attributes.

```
interface warehouse
{
read_warehouse_record([comm_status,fault_status]st);
ship_and_bill([comm_status, fault_status] st);
}
```

File 7. `warehouse.acf`: Warehouse interface additional attributes.

IDL COMPILATION AND STUBS

Once the IDL and ACF files for an interface are written, they are fed into a DCE tool called the *IDL compiler* (`idl`). The IDL compiler checks for correct IDL syntax and semantics and produces a series of C language source files. For an interface named `X`, the IDL compile produces three files: the interface header file

X.h, the client stub file X_cstub.c, and the server stub file X_sstub.c. The stub files must then be compiled with the POSIX C compiler (c89) to produce object files. The IDL and C compilations must be done in the POSIX shell. The process is diagrammed in Figure 3.

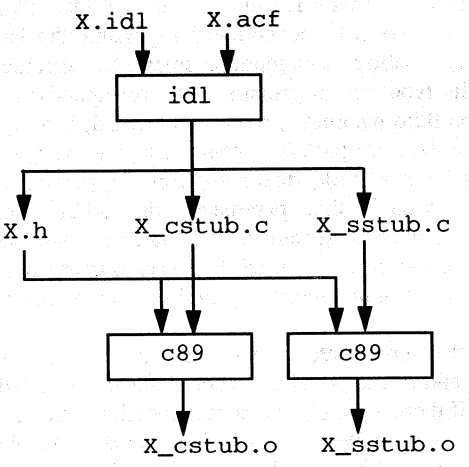


Figure 3. Stub Compilation for interface X.

The header file contains definitions of data structures used by the stub files and by the client and server programs. One example is the *interface specification*, which encodes the interface uuid, version, number of remote procedures, and other relevant information. The client and server stub files contain the code for one *stub procedure* for each remote procedure in the interface. These stub procedures are used by RPC to perform the remote call. Figure 4 shows the actions of the stubs, using the read_stock_record remote procedure. The steps are as follows (ignoring the handle and status parameters):

1) CLIENT calls read_stock_record, passing in stock_number.
2) read_stock_record (actually the client stub procedure) marshals the input parameter stock_number into a request packet and calls the RPC Runtime.
3) Client RPC Runtime sends the request packet to the server RPC Runtime .
4) Server RPC Runtime receives the request packet and calls the server stub procedure op0_ssr.
5) op0_ssr unmarshals the input parameter stock_number and calls read_stock_record (the target procedure) passing in stock_number.
6) read_stock_record executes and returns to op0_ssr, passing back stock_record.

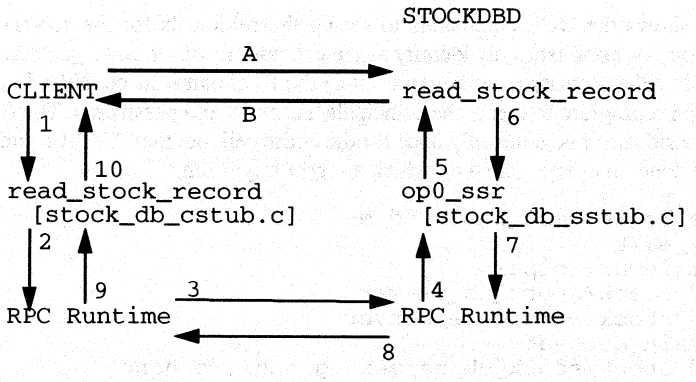


Figure 4. Stub Operation in a Remote Procedure Call.

7) `op0_ssr` marshals the output parameter `stock_record` into a response packet and returns to the server RPC Runtime.
8) Server RPC Runtime sends the response packet back to the client RPC Runtime.
9) Client RPC Runtime receives packet and returns to `read_stock_record` (client stub).
10) `read_stock_record` unmarshals the output parameter `stock_record` and passes it back to the client.

The apparent operation is:
A) `CLIENT` calls `read_stock_record`, passing in `stock_number`.
B) `read_stock_record` executes and returns `stock_record`.

CLIENT-SERVER DESIGN: GENERAL CONCEPTS

Now that I have written and compiled the interfaces for the application, I can start to look at the actual client and server programs. But first I will discuss a couple of DCE concepts relating to security and the RPC name space. I will also preview my approach to interfacing with DCE services from COBOL.

Security Principals

I need to set up some security principals, with passwords, to be used by the clients and servers. Each type of server will have its own principal. For example, `STOCKDBD` will use the principal `stock/stock_db_server`. When a server starts up, it will perform a programmatic login using its principal and password. The clerks using the `CLIENT` program will also be assigned principals.

File 8a shows the DCE commands to set up the principals for the servers. The `dce_login` establishes my identity as the cell administrator `cell_admin`, so I can modify the security database and (later) the RPC name space. Next I use the `rgy_edit` program to create the principals, accounts and passwords. The full set of commands need be done only once for the entire cell, but the `ktadd` commands must be done on every system on which a server might run.

```
dce_login cell_admin -dce-
rgy_edit
domain principal
add stock/stock_db_server
add stock/warehouse_server
domain account
add stock/stock_db_server -g none -o none
    -pw stock_db_password -mp -dce-
add stock/warehouse_server -g none -o none
    -pw warehouse_password -mp -dce-
ktadd -p stock/stock_db_server
    -pw stock_db_password -f /krb5/v5srvtab
ktadd -p stock/warehouse_server
    -pw warehouse_password -f /krb5/v5srvtab
quit
```

File 8a. `configure.sh`: Security configuration.

RPC Name Space Entries

Next I need to create some RPC name space entries that the servers will use for their registration, and the client will use for its lookup. I will use a two level scheme[6]. Each server will have its own *RPC entry*, which will hold the server's location and protocol binding information. All the servers of the same type will share a *RPC group entry*, which will hold the names of the RPC entries for the servers of that type. The registration and lookup code given later will use these entries.

File 8b shows the commands to set up the name space entries. The `cdscp` command creates the directory `/.:/stock` in CDS. The `rpccp` commands create the RPC entries and group entries for one stock and two warehouse servers. The `acl_edit` commands set up access control lists for the entries. These ACLs allow servers of the appropriate type to modify their entries and protect the entries against surreptitious tampering by unauthorized principals.

6. This naming scheme (and the DCE code to use it) has been taken from the OSF DCE Demographic Demonstration Program (DDP), which was developed by the University of Michigan Center for Information Technology Integration. DDP was one of the principal demos shown at the OSF Challenge '93 event.

```
cdscp create directory /.:/stock
rpccp add entry /.:/stock/stock_db.grp
rpccp add entry /.:/stock/stock_db_1
rpccp add entry /.:/stock/warehouse.grp
rpccp add entry /.:/stock/warehouse_1
rpccp add entry /.:/stock/warehouse_2
acl_edit -e /.:/stock/stock_db.grp
   -m user:stock/stock_db_server:rwdtc
acl_edit -e /.:/stock/stock_db_1
   -m user:stock/stock_db_server:rwdtc
acl_edit -e /.:/stock/warehouse.grp
   -m user:stock/warehouse_server:rwdtc
acl_edit -e /.:/stock/warehouse_1
   -m user:stock/warehouse_server:rwdtc
acl_edit -e /.:/stock/warehouse_2
   -m user:stock/warehouse_server:rwdtc
```

File 8b: `configure.sh`: Name space entry configuration.

Setup Library and Interface-Specific Setup Routines

Before I can write the client and server programs in COBOL, I must address two major problems:

1. Many DCE services are effectively callable only from C. They use data types and definitions from DCE include files in C.
2. Clients and servers need to use interface-specific information from the C include file generated by the IDL compiler.

The approach I will to take to solve these problems is:

1. Most of the code needed to set up the clients and servers will be collected into a library of C source, called `dce_setup_lib.c`. This code will be generic enough that it can be used by all the clients and servers I might want to write. It will include procedures to perform the security login, to register and lookup interfaces in the RPC name space, and various other DCE housekeeping functions.

2. For each interface I will generate specific C source modules from a template. These interface-specific files will include the header file for the interface and will call the generic setup routines in `dce_setup_lib.c`. The routines in the interface files will be directly callable from COBOL; they will not require any C specific data types or values. Following the stub model, each interface will have two files: *INTERFACE*`_v`*VERSION*`_csetup.c` and *INTERFACE*`_v`*VERSION*`_ssetup.c`, where *INTERFACE* is the name of the interface and *VERSION* is the interface version. (The inclusion of the interface version permits

concurrent usage of more than one interface version in a client or server.) For the `stock_db` interface the files would be `stock_db_v1_0_csetup.c` and `stock_db_v1_0_ssetup.c`. To generate these files, I will edit the template files (with the POSIX `vi` editor) and substitute the interface name, version, and the procedure name list. Once the interface files are generated, they are compiled using the POSIX C compiler. These actions are diagrammed in Figure 5.

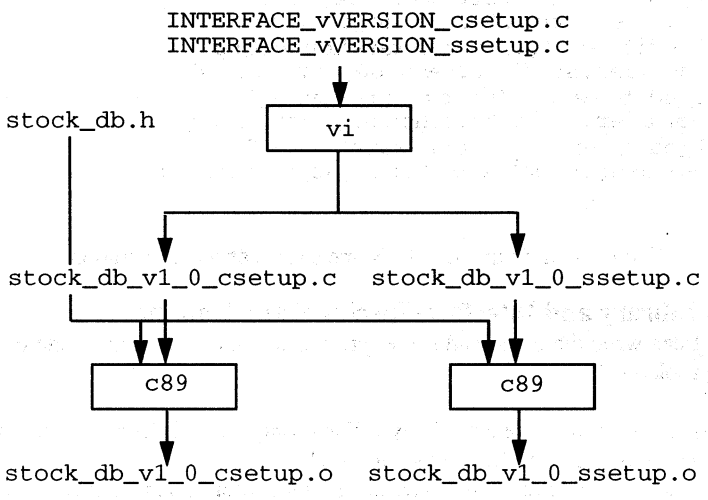


Figure 5. Generation of interface-specific setup files from templates.

The value of this approach is that once the setup library and templates are developed, they can be reused for any client and server in COBOL (or any other language, for that matter). The application developer need not have detailed DCE and C programming knowledge.

SERVER DESIGN

Now I can begin the actual design of the server. First I will cover the security and setup routines that go into the setup library, followed by the interface-specific routines. Finally I will discuss the actual COBOL servers.

Server Security

A server will call the routine `dce_security_login` to establish its principal identity with DCE Security, passing in the principal and password as simple character strings. `dce_security_login` calls DCE Security service routines

to setup the principal identity, validate the password, certify that the Security service is genuine, and set up the login context. It also calls the RPC routine to set up authentication for the server. File 9a shows the routine.[7]

```
void dce_security_login(principal, password)
unsigned_char_t *principal;
unsigned_char_t *password;
{
   ret = sec_login_setup_identity(...);
   ret = sec_login_validate_identity(...);
   sec_login_certify_identity(...);
   rpc_server_register_auth_info(...);
}
```

File 9a. `dce_setup_lib.c: dce_security_login`

Server Setup

Server setup includes the following tasks:

1. Register the interface with the RPC Runtime.
2. Obtain sockets for the desired protocol sequences.
3. Obtain bindings for those sockets.
4. Register the interface and bindings with the RPC Endpoint Map.
5. Export the interface and bindings to the RPC Name Space entry for the server.
6. Add the RPC server entry to the RPC group entry for the server type.
7. Set a management authorization function to allow certain RPC management functions to be performed.

The routine `dce_setup_group_interface` encapsulates the calls for these tasks. It is shown in File 9b. The input to this routine includes the interface specification, manager entry point vector for the remote procedures to be called, the name space entry and group names, and the protocol sequence to be used. Output from the routine is a vector of bindings created for the interface.

A COBOL program cannot directly call `dce_setup_group_interface`, because it cannot pass in the interface specification (which comes from the interface's header file) or the manager entry point vector. So I will have to create an interface-specific routine, `setup_group_`*`INTERFACE`*`_v`*`VERSION`*, to call `dce_setup_group_interface` with the parameters for an interface. This routine will reside in the server setup template and will be edited to insert the actual interface name, version, and procedure list. Figure 10a shows the routine.

7. For the sake of brevity, only outlines of the C code will be shown. Complete listings may be requested from the author at the address at the beginning of this paper.

```
void dce_setup_group_interface(ifspec, manager_epv,
     ns_entry, ns_group, protseq, binding_vector);
rpc_if_handle_t ifspec;
rpc_mgr_epv_t manager_epv;
unsigned_char_t *ns_entry;
unsigned_char_t *ns_group;
unsigned_char_t *protseq;
rpc_binding_vector_t **binding_vector;
{
   rpc_server_register_if(...);
   if (*protseq_ == '\0')
      rpc_server_use_all_protseqs(...);
   else
      rpc_server_use_protseq(...);
   rpc_server_inq_bindings(...);
   rpc_ns_binding_export(...);
   rpc_ns_group_mbr_add(...);
   rpc_mgmt_set_authorization_fn(...);
}
```

File 9b. dce_setup_lib.c: dce_setup_group_interface

```
#include "INTERFACE.h"

rpc_binding_vector_t
     *INTERFACE_vVERSION_binding_vector;
INTERFACE_vVERSION_epv_t INTERFACE_vVERSION_epv =
     {PROCEDURE1, PROCEDURE2, ...}

void setup_group_INTERFACE_vVERSION(ns_entry,
                                    ns_group, protseq)
unsigned_char_t *ns_entry;
unsigned_char_t *ns_group;
unsigned_char_t *protseq;
{
   dce_setup_group_interface(
      INTERFACE_vVERSION_s_ifspec,
      (rpc_mgr_epv_t *) &INTERFACE_vVERSION_epv,
      ns_entry,
      ns_group,
      protseq,
      &INTERFACE_vVERSION_binding_vector);
}
```

File 10a. INTERFACE_vVERSION_ssetup.c: setup_group_...

Server Cleanup

When a server terminates, there are some actions it should perform to clean up after itself:

1. Remove ("unregister") the interface and bindings from the RPC Endpoint Map.
2. Remove ("unexport") the interface and bindings from the RPC Name Space entry for the server.
6. Remove the RPC server entry from the RPC group entry for the server type.
7. Deallocate the binding vector for the interface bindings.

The routine `dce_cleanup_group_interface`, shown in File 9c, performs these operations. It requires as input the interface specification from the interface header file, the binding vector returned the `dce_setup_group_interface` routine, and the name space server and group entry names. As with the setup routine, the cleanup routine cannot be called directly from COBOL. Instead, there is a customizable routine `cleanup_group_`*`INTERFACE_vVERSION`* in the server setup template file that fills in the interface-specific parameters. This is shown in File 10b.

```
void dce_cleanup_group_interface(ifspec,
     binding_vector, ns_entry, ns_group);
rpc_if_handle_t ifspec;
rpc_binding_vector_t **binding_vector;
unsigned_char_t *ns_entry;
unsigned_char_t *ns_group;
{
   rpc_ep_unregister(...);
   rpc_ns_binding_unexport(...);
   rpc_ns_group_mbr_remove(...);
   rpc_binding_vector_free(...);
}
```

File 9c. `dce_setup_lib.c: dce_cleanup_group_interface.`

```
void cleanup_group_INTERFACE_vVERSION(ns_entry,
                                      ns_group)
unsigned_char_t *ns_entry;
unsigned_char_t *ns_group;
{
   dce_cleanup_group_interface(
      INTERFACE_vVERSION_s_ifspec,
      &INTERFACE_vVERSION_binding_vector,
      ns_entry, ns_group);
}
```

File 10b. `INTERFACE_vVERSION_ssetup.c: cleanup_group_...`

Server Main

Now that I have all of setup code written in C, I can finally write my servers in COBOL. The servers will consist of a main part and the manager routines. The server main calls the security and interface-specific setup routines and any application-specific initialization like opening databases. It then calls an RPC routine `rpc_server_listen`, which enters the RPC Runtime for the duration of the server's existence. `rpc_server_listen` will return to the main program only if the server is stopped by RPC -- for example, by a client that calls the RPC management routine `rpc_mgmt_stop_server_listening` with a binding handle for the server. So, following `rpc_server_listen`, the server will call the interface-specific cleanup routine and any application-specific cleanup, like closing databases.[8]

```
$CONTROL USLINIT,SUBPROGRAM
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 PRINCIPAL  PIC X(25) VALUE "stock/stock_db_server ".
01 PASSWORD   PIC X(18) VALUE "stock_db_password ".
01 NS-ENTRY   PIC X(24) VALUE "/.:/stock/stock_db_1 ".
01 NS-GROUP   PIC X(28) VALUE "/.:/stock/stock_db.grp ".
01 PROTSEQ    PIC X(13) VALUE "ncacn_ip_tcp ".
01 STAT       PIC S9(9) COMP.
. . .
LINKAGE SECTION.
. . .
PROCEDURE DIVISION.
BEGIN
    STOP RUN.
. . .
ENTRY "MAIN"
    CALL "DCE_SECURITY_LOGIN" USING PRINCIPAL PASSWORD.
    CALL "SETUP_GROUP_STOCK_DB_V1_0" USING
         NS-ENTRY NS-GROUP PROTSEQ.
    PERFORM OPEN-STOCK-DB.
    CALL "RPC_SERVER_LISTEN" USING \1\ STAT.
    CALL "CLEANUP_GROUP_STOCK_DB_V1_0" USING
         NS-ENTRY NS-GROUP.
    PERFORM CLOSE-STOCK-DB.
    STOP RUN.
. . .
```

File 11a. `STDBDCOB`: Stock DB Server Main.

8. At this point, the astute reader might ask, since most of the work in the server main is done by the C setup/cleanup routines, why not write the server main in C as well? The principal reason is to preserve the application-specific initialization and cleanup code from the original COBOL application.

File 11a shows the STOCKDBD main program. In the WORKING-STORAGE SECTION I've set up the data items to be passed to the security and setup routines. These include the server's principal name and password, the name space server entry and group entry names, and the protocol sequence to use. All of these data items are simple character strings, but note that all are terminated by a blank. Normally C character strings are terminated by a NULL (0) character, but COBOL does not easily allow that. So I've adopted the convention that all character strings passed into the dce_setup_lib.c routines must be terminated by a blank. Those routines are then responsible for converting the strings to NULL-terminated strings for C usage.

Note well that the main part of the PROCEDURE DIVISION has turned into a procedure named main, by prefixing it with ENTRY "MAIN". This un-COBOL-like requirement comes from the POSIX C library (/lib/libc.a) that must be linked with any program using DCE. The C library expects to find a main routine that it calls during process startup.

The main program calls the set up routines dce_security_login, setup_-group_stock_db_v1_0, and the cleanup routine cleanup_group_-stock_db_v1_0 routines, using the statement CALL "*PROCEDURE_NAME*" USING *P1 P2 ...*, where *P1 P2 ...*, are the procedure parameters.[9] An important point to remember is that all parameters are normally passed by reference, so all parameters will be pointers to the data values. This is what we want for character strings. For numeric values we have to be careful.

After performing the security login and setting up the stock_db interface, the main program performs the OPEN-STOCK-DB paragraph to open the stock database. This paragraph can be taken from the original COBOL application. Now that all initialization is complete, the server calls the RPC routine rpc_server_listen. This routine has one input parameter -- the number of calls that the server can concurrently handle. Within RPC this becomes the number of threads that can concurrently execute the server's remote procedures. The subject of threads is beyond the scope of this paper. Suffice it to say that using more than one call thread greatly complicates the design of the remote procedures. You may need to use such esoteric facilities as thread mutex locking to protect global data within the server. My strong recommendation is to use one call thread, which is the parameter \1\ to rpc_server_listen.[10] The second parameter is the return status from rpc_server_listen, which I will ignore here.

---

9. The procedures names are coded in lowercase in C and in uppercase in COBOL.
10. The \ \ allows the parameter value to be passed, instead of an address.

Server Manager Procedures

The last things to be added to the server program are the actual bodies of the remote procedures. These are called *manager* routines. Frequently manager routines are placed in a separate compile module. But in my example, I have put them in with the main program. Each manager routine is an entry point, declared with the statement ENTRY "*PROCEDURE NAME*" USING *P1 P2 ...*, where *P1, P2, ...* are the procedure's parameters. These parameters are declared in the LINKAGE SECTION of the DATA DIVISION. Remember once again that all parameters in COBOL are passed by reference. The body of the procedure follows the ENTRY statement. In the body there must be one or more GOBACK statements to return control to the caller -- in this case, the server executing the procedure's server stub.

File 11b shows the manager procedure READ_STOCK_RECORD for STOCKDBD. The parameters in the ENTRY statement match those in stock_db.idl for read_stock_record: the binding handle H (which is not used in the manager), the input stock number ST-NUMBER, the output record STOCK-RECORD, and the return status ST. The body of the procedure, which was taken from the original READ-STOCK-RECORD paragraph, is omitted here. At the succesful completion of the procedure there are statements to set ST to the ERROR-OK status and to return to the caller.

```
$CONTROL USLINIT,SUBPROGRAM
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
. . .
01 ERROR-OK    PIC S9(9) COMP VALUE 0.
LINKAGE-SECTION.
01 H           PIC S9(9) COMP.
01 ST-NUMBER   PIC X(10).
$INCLUDE RECTCOB
01 ST          PIC S9(9) COMP.
. . .
PROCEDURE DIVISION.
BEGIN.
   STOP RUN.
ENTRY "READ_STOCK_RECORD" USING
        H ST-NUMBER STOCK-RECORD ST.
   . . .
   MOVE ERROR-OK TO ST.
   GOBACK.
ENTRY "MAIN"
   . . .
```

File 11b. STDBDCOB: Stock Server Manager Procedure

Server Compilation and Linkage

To compile the server program file STDBCOB, I used the COBOL 85 compiler, `cob85xl`. This command is executed from the MPE CI, takes as input a normal MPE ASCII file (that is, not a POSIX byte stream file), and produces an MPE object file. To build the server program, I linked the following object files and libraries:

`STDBO`: server main and manager procedures
`stock_db_sstub.o`: stock_db interface server stub routines
`stock_db_v1_0_ssetup.o`: stock_db interface server setup routines
`dce_setup_lib.o`: generic DCE setup routines
`/lib/libdce.a`: DCE Runtime Library
`/lib/libsock.a`: Socket Library

The actual compile and link scripts are given the Appendix.

The `WHOUSED` server program is very similar to `STOCKDBD`. It has a server main source in COBOL which uses the relevant routines for the warehouse interface, and which includes manager routines for `READ_WAREHOUSE_RECORD` and `SHIP_AND_BILL`. That code is omitted here.

CLIENT DESIGN

Lastly I can design the client program. First I will cover the client side routines that go into `dce_setup_lib.c` and interface-specific client setup routines. Then I will discuss the actual remote calls and the final `CLIENT` program in COBOL.

Client Security

Unlike the servers, the client will assume the identity of the order clerk that runs it. The clerk will execute a `dcelogin` *`principal password`* command before running `CLIENT`. The clerk's principal, and its password, would be assigned by the cell admininstrator using `rgy_edit`. The cell admininstrator may want to create a security group, say `order_clerk`, to which all of the order clerk principals belong. That way ACLs to protect the stock and warehouse databases can be set up for all of the order clerks. The actual method will not be covered in this paper.

The client calls the routine `dce_get_users_login`, shown in File 9d, to inherit the clerk's login. This routine simply calls the DCE Security call to get the current login context.

```
sec_login_handle_t login_context;
. . .
void dce_get_users_login()
{
   . . .
   sec_login_get_current_context(...);
}
```

File 9d. `dce_setup_lib.c: dce_get_users_login`

Client Binding

Before it can make any remote calls, the client must find a server to use. This action is called *binding*. During binding, RPC creates a *binding handle* to be used in the remote calls -- this is the first parameter `h` for the remote procedures. The binding handle contains the *network address* to use to communicate with the server and the state of that communication. There are a number of ways to set up a binding handle in DCE, depending on how much of the binding information is known to the client. In the example I have used the most general method: registration and lookup of the binding information in the RPC Name Space. I discussed the RPC server and group entries used for this purpose earlier.

The `dce_bind_group_interface` routine, shown in File 9e, sets up one or more binding handles for a group of servers. The input parameters are the specification of the server's interface, the server's group name, the level of protection to be used in the remote calls (from no protection to encryption), and the maximum number bindings that can be returned to the caller. The output parameters are an array (a *table* in COBOL terminology) of binding handles and the number of binding handles returned. There will be one binding handle for each active server registered in the server group. The client program can then choose which servers it wants to use.

There are three calls that `dce_bind_group_interface` makes to obtain the binding handles. The first call is to `rpc_ns_binding_import_begin` to specify the server group and interface specification. This returns an *import context* which is used in a series of `rpc_ns_binding_import_next` calls to get the registered bindings from the name space. Each binding is tested to see if the server is still active. (It's possible that the server could have aborted without properly cleaning up the name space.) Also, the desired security level is set for each good binding. After all of the bindings have been extracted from the name space, `rpc_ns_binding_import_done` frees the import context.

The interface-specific routine to call dce_bind_group_interface is shown in File 12. This is in the template file `INTERFACE_vVERSION_csetup.c`

```
void dce_bind_group_interface(ifspec, ns_group,
     protect_level, max_handles, handles, n_handles);
rpc_if_handle_t          ifspec,
unsigned_char_t          *ns_group;
unsigned32               protect_level;
long int                 max_handles;
rpc_binding_handle_t     handles[];
long int                 *n_handles;
{
   . . .
   rpc_ns_binding_import_begin(...);
   for (i=0; i < max_handles; i++)
   {
      rpc_ns_binding_import_next(...);
      if (status == rpc_s_no_more_bindings)
         break;
      listening = rpc_mgmt_is_server_listening(...);
      . . .
      if (!listening)
      {
         i--;
         continue;
      }
      rpc_binding_set_auth_info(...);
   }
   rpc_ns_binding_import_done(...);
   . . .
}
```

File 9e. dce_setup_lib.c: dce_bind_group_interface

```
#include "INTERFACE.h"
void bind_group_INTERFACE_vVERSION(ns_group,
     protect_level, max_handles, handles, n_handles);
unsigned_char_t          *ns_group;
unsigned32               protect_level;
long int                 max_handles;
rpc_binding_handle_t     handles;
long int                 *n_handles;
{
   dce_bind_group_interface(
      INTERFACE_vVERSION_s_ifspec,
      ns_group,
      protect_level,
      max_handles,
      handles,
      n_handles);
}
```

File 12. INTERFACE_vVERSION_csetup.c

Client Main Program

The CLIENT main program source is shown in File 13a. In the WORKING-STORAGE SECTION of the DATA DIVISION are declared the data items to be used in binding to the servers, including the server group names, the desired protection level (*packet integrity*, which ensures that the remote call packets are not tampered with), and the tables to hold up to ten bindings for each server group. As with the servers, the client program must have a `main` procedure because of the requirements of the POSIX C library. This includes calls to set up security and to bind to the stock_db and warehouse servers. The OPEN-TERMINAL and CLOSE-TERMINAL paragraphs are preserved from the original application.

```
$CONTROL USLINIT,SUBPROGRAM
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 STOCK-GRP  PIC X(24) VALUE "/.:/stock/stock_db.grp ".
01 WHOUSE-GRP PIC X(25) VALUE "/.:/stock/warehouse.grp".
01 PKT-INTEG  PIC S9(9) COMP VALUE 5.
01 MAX-H      PIC S9(9) COMP VALUE 10.
01 N-STOCK-H  PIC S9(9) COMP.
01 N-WHOUSE-H PIC S9(9) COMP.
01 STOCK-BINDINGS
  02 STOCK-H  OCCURS 10 TIMES PIC S9(9) COMP.
01 WHOUSE-BINDINGS
  02 WHOUSE-H OCCURS 10 TIMES PIC S9(9) COMP.
. . .
PROCEDURE DIVISION.
BEGIN
    STOP RUN.
ENTRY "MAIN"
    CALL "DCE_GET_USERS_LOGIN".
    CALL "BIND_GROUP_STOCK_DB_V1_0" USING
         STOCK-GRP \PKT-INTEG\ \MAX-H\
         STOCK-BINDINGS N-STOCK-H.
    IF N-STOCK-H = 0 THEN
         STOP RUN.
    CALL "BIND_GROUP_WAREHOUSE_V1_0" USING
         WHOUSE-GRP \PKT-INTEG\ \MAX-H\
         WHOUSE-BINDINGS N-WHOUSE-H.
    IF N-WHOUSE-H = 0 THEN
         STOP RUN.
    PERFORM OPEN-TERMINAL.
    PERFORM PROCESS-TRANSACTION UNTIL EXIT = "YES".
    PERFORM CLOSE-TERMINAL.
    STOP RUN.
```

File 13a. CLNTCOB: Client main program.

Client Remote Procedure Calls

At long last, the client can make the remote procedure calls. These are in the PROCESS-TRANSACTION paragraph (File 13b), where they have replaced some of the PERFORMs from the original application. The first parameter of each remote call is the server binding handle. For simplicity, I have chosen to use the first values returned in the binding handle tables. In a real application I would want to select the binding handle (and thus the server) based on more complex criteria, such as the location of the warehouse. Most of the other parameters to the remote procedures are data items or records defined in the RECTCOB data definition file.

Unlike local procedure calls, remote procedure calls can fail because the communications to the server failed or because the server itself failed. The client should be aware of when this happens and take appropriate action. There are two ways the client can detect these errors. The first way is to use the DCE exception handling facilities. In a C program, I would wrap TRY/CATCH/ENDTRY exception handling macros around the call. However, since these are C macros, they are not available for COBOL programs. The other method of handling errors is to add an explicit status parameter, designated for fault and communications status in the interface ACF file. When a status parameter is specified, the IDL compiler will insert the TRY/CATCH/ENDTRY macros into the client stub routine, which are in C. The client stub routine will turn any detected exception into a return status that can be inspected by the calling program. I strongly recommend that all remote procedure calls from COBOL programs use this facility. The last parameter of each remote call in the CLIENT is such a status parameter. If the return status is not ERROR-OK, the client will perform the RPC-ERROR paragraph. This paragraph calls dce_error_inq_text to translate the status into a message, displays the message, and stops the program. In a more complete application, I might want to retry the failed remote call with another binding handle to a different server.

Client Linkage

The following object files and libraries are linked to produce the CLIENT program:

CLNTO: client main
stock_db_cstub.o: stock_db interface client stub routines
stock_db_v1_0_csetup.o: stock_db interface client setup routines
warehouse_cstub.o: warehouse interface client stub routines
warehouse_v1_0_csetup.o: warehouse interface client setup routines
dce_setup_lib.o: generic DCE setup routines
/lib/libdce.a: DCE Runtime Library
/lib/libsock.a: Socket Library

```
PROCEDURE DIVISION.
ENTRY "MAIN"
. . .
PROCESS-TRANSACTION.
    PERFORM INPUT-TRANSACTION.
    CALL "READ_STOCK_RECORD" USING
          STOCK-H(1) STOCK-NUMBER OF TRANSACTION-RECORD
          STOCK-RECORD ST.
    IF ST <> ERROR-OK THEN
          PERFORM RPC-ERROR.
    CALL "READ_WAREHOUSE_RECORD" USING
          WHOUSE-H(1) STOCK-NUMBER OF TRANSACTION-RECORD
          WAREHOUSE-RECORD ST.
    IF ST <> ERROR-OK THEN
          PERFORM RPC-ERROR.
    IF QTY-ON-HAND OF WAREHOUSE-RECORD <
       QTY-TO-SHIP OF TRANSACTION-RECORD THEN
       PERFORM NOT-ENOUGH-STOCK
    ELSE
       CALL "SHIP-AND-BILL" USING
            WHOUSE-H(1) STOCK-RECORD
            TRANSACTION-RECORD ST
       IF ST <> ERROR-OK THEN
            PERFORM RPC-ERROR.
. . .
INPUT_TRANSACTION.
. . .
RPC-ERROR.
    CALL "DCE_ERROR_INQ_TEXT" USING \ST\ ERROR-TEXT ST1.
    PERFORM DISPLAY-ERROR-TEXT.
    STOP RUN.
```

File 13b. CLNTCOB: Client remote procedure calls.

POSSIBLE DCE ENHANCEMENTS FOR COBOL

All the techniques I have covered so far in this paper can be done now with the existing DCE tools. As you can see, these guidelines make it possible to use DCE from COBOL. However, you can also see that the some of the interface and client/server design tasks will not be easy for COBOL programmers. Here I will take a look at some possible enhancements to DCE to increase its ease of use from COBOL. These enhancements are still suggestions now, with no commitments from HP or OSF. If you like these ideas (or have ideas of your own) you should let

us know.

COBOL-style IDL

Perhaps the most unnatural thing DCE forces a COBOL programmer to do is to write C-style interface definitions. A COBOL flavor of IDL would allow the programmer to write data definitions in a more familar syntax. A COBOL IDL file might look like File 14.

```
UUID 7E3F8A8E-5538-11CC-A91A-080009226619.
VERSION 1.0.
INTERFACE STOCK_DB.

DATA-DIVISION.
LINKAGE-SECTION.
01 H            HANDLE.
01 ST-NUMBER    PIC X(10).
$INCLUDE RECTCOB
01 ST           PIC S9(9) COMP.

PROCEDURE DIVISION.
ENTRY "READ_STOCK_RECORD" USING
       [IN] H
       [IN] ST-NUMBER
       [OUT] STOCK-RECORD
       [OUT] ST.
```

File 14. A possible COBOL style IDL definition.

COBOL-callable DCE Services

Much of the work in this paper was to get around the problem that the DCE Services API is not directly callable from COBOL. A DCE COBOL API would allow the COBOL programmer more flexibility in using DCE services. This API would need COBOL include files defining DCE data types and values.

ENCINA

Encina is set of tools from Transarc Corporation for developing transaction processing applications on top of DCE. Like DCE, Encina will soon be available on a number of platforms from HP, IBM, Sun, and so on. Since transaction integrity is important in commercial applications, many customers will want to use Encina to develop their applications. Encina, like DCE, is designed to be used from C, so COBOL programmers will face many of the same problems using Encina. A full discussion of Encina is beyond the scope of this paper. But many of the techniques described here can be applied to Encina.

Transactional RPC and TIDL

Encina extends the remote procedure call facility of DCE to include transaction semantics. It provides a *Transaction Interface Definition Language (TIDL)* to define transaction interfaces and a TIDL compiler to produce stub routines and IDL files. TIDL is very similar to IDL, so the discussion on COBOL to IDL translation is applicable to TIDL as well.

Encina Services

Like the DCE services, Encina services are provided by C-callable routines. And like the DCE services, they can be encapsulated in generic and interface-specific modules. COBOL clients and servers can (in theory, at least) call the encapsulation routines to use Encina. More research is needed to determine the best methods of encapsulation.

APPENDIX: EXAMPLE APPLICATION BUILD SCRIPTS

There are two scripts to build the example application. The first one is executed from the POSIX shell. It does the IDL and C compilations. The second script is executed from the MPE CI. It does the COBOL compilations and linkages. These scripts assume an MPE group and account `STOCK.HPDCE` is used for the MPE files, and POSIX directories `src` and `obj` exist within the group for POSIX files.

```
IDL_FLAGS="no_cpp keep c_source"
cd /HPDCE/STOCK/obj
rm *
idl $IDL_FLAGS ../src/record_types.idl
idl $IDL_FLAGS -I$S ../src/stock_db.idl
c89 -c -I. -DMPEXL stock_db_cstub.c
c89 -c -I. -DMPEXL stock_db_sstub.c
idl $IDL_FLAGS -I$S ../src/warehouse.idl
c89 -c -I. -DMPEXL warehouse_cstub.c
c89 -c -I. -DMPEXL warehouse_sstub.c
c89 -c ../src/dce_setup_lib.c
c89 -c -I. ../src/stock_db_v1_0_csetup.c
c89 -c -I. ../src/stock_db_v1_0_ssetup.c
c89 -c -I. ../src/warehouse_v1_0_csetup.c
c89 -c -I. ../src/warehouse_v1_0_ssetup.c
```

File A1. `make.sh`: POSIX script for IDL and C compilation

```
cob85xl stdbcob,stdbo,$null
link from=stdbdo,./obj/dce_setup_lib.o, &
     ./obj/stock_db_v1_0_ssetup.o, &
     ./obj/stock_db_sstub.o; &
     to=./STOCKDBD; &
     rl=/lib/libdce.a,/lib/libsock.a
```

File A2. MKSTDB: MPE script for COBOL compilation and linkage.

REFERENCES

The following is the standard DCE manual set available from:

Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142

Introduction to OSF DCE.
OSF DCE Version 1.0, DCE User's Guide and Reference.
OSF DCE Version 1.0, DCE Porting and Testing Guide.
OSF DCE Version 1.0, DCE Application Development Guide.
OSF DCE Version 1.0, DCE Administration Guide
OSF DCE Version 1.0, DCE Application Development Reference.
OSF DCE Version 1.0, DCE Administration Reference.

Paper # 7010

Project Management and 4GL Methodologies

by

Suzanne Harmon
Information Systems Consultant
(408) 459-0802

Project management is always a challenge at best and can be a disaster at worst. As more and more Information Systems groups move to fourth generation development environments, they are finding that their old project management strategies are no longer effective, and yet they are not equiped with new methodologies to replace them. Many new techniques such as entity relationship modelling, data flow diagrams, and CASE management provide tools for design in these new environments, but little or no attention is given to the new management techniques required to optimize the benefits of moving to this new technology.

Over many years of managing 4GL development projects within my own company I evolved a project management methodology which worked as close to perfectly as one could believe possible. After leaving the company, I was examining my collection of skills trying to decide where I wanted to go professionally, and thought I could bring real value to organizations by training them in this methodology, essentially porting it to their environment. This process has not had the success I had hoped it would, and that has perplexed me and caused me great consternation. Then a few months ago I was having a discussion with a close friend who works with the object oriented data base team at Hewlett-Packard about the work he was doing. I was astonished to learn that not only was he using the same management techniques which I had evolved, but that there is a growing movement afoot in selected corners of corporate America, as well as Harvard Business School, towards this method of management. My friend gave me many articles he had collected and one he had written which discuss this management process and how it has been used by various organizations. What really interested me as regards what I percieve as my failure to

port this methodology into other organizations were a couple of quotes in one of the articles in Fortune Magazine, May 18, 1992 entitled "Looking Ahead." "According to Herman Simon ... higher ups who saw the numbers vowed never to mess with the plant. But they rarely went away determined to make their other factories over in its image."
And, "Says a frustrated William Buehler, senior vice president at Xerox, 'You can see a high performance factory or office, but it just doesn't spread, I don't know why.'" This was definitely my experience. I could site project histories which proved the methodology and sell the concept, but when it came to actually implementing it at the customer's site I could not get co-operation, support or commitment from either above or below. Part of this, I feel, is a syndrome which seems to be typical to MIS organizations if not all organizations. They want to give lip service to incorporating or espousing "in" methodologies and technologies. They buy expensive CASE products, and send their employees to expensive classes about Entity Relationship Modelling. It impresses the higher ups and the auditors with MIS Management's commitment to being leading edge and up to date. But, usually, lip service is about where it ends, as most people are very resistent to changing the status quo except very slowly and over a long period of time.

As we discuss this methodology, I can give you hows and whys, but I cannot tell you what must be done within an organization to get the support and commitment needed to effect the kind of change we're talking about for the average organization. I will refer to this methodology as 4GL Project Management and Development because that is how it evolved for me, but you will see if you pursue some of the other articles I refer to, that it is really more far reaching that that, and can be used with variation to manage any organization or process.

We are going to discuss this managment as it relates to what I consider to be the four phases of a new system development project. These phases are the estimating, the planning, the development, and the implementation readiness which includes quality control, documentation and training. However, I am not going to discuss them in that order (which is as they occur) because it is easier to understand what is going on in planning and estimating if you already understand a bit about the development process itself.

The Development

The hardest part of transitioning from 3GL projects to 4GL projects, is getting people to let go of their individual egos and create a team ego. The key to successful 4GL development projects is the team dynamic. Every project team which consists of more than one person is a combination of people who are not all equal in their skills, their competence, or their productivity. The individual skills or lack thereof are only important as they affect the composite performance of the team. People who insist on maintaining their individual ego as a member of the team get in the way of the team's ability to acheive its optimum dynamic.

The entire team meets weekly, at the same time every week. It is at these meetings that assignments are made by the project manager which will insure that the next milestone is met. Assignments are only rarely made which do not directly bear on the upcoming milestone. In this way, the entire team keeps focused on the goal. As assignments are handed out, taking into consideration peoples skills and productivity, the individual receiving the assignment needs to be queried to insure that they feel they can meet the commitment being asked of them - to have this work being assigned completed to a specified degree in a specified timeframe. If people make commitments that they cannot realistically meet, they are letting down the whole team. This is in fact a major learning process for most team members unfamiliar with this style of management, because most people are accustomed to overcommitting and really have no true idea of what they can, in fact, accomplish over a given period of time. Ideally, the amount assigned to each person is never as much or more than they can accomplish, for two reasons. The first is that people need to be successful and not get into habits of not completing what is expected of them. From a project perspective, the second reason is the most important, which is to insure that people have room in their schedules to add unexpected tasks that come up or to complete work, assigned to another person, which is behind schedule. And finally, each person must have a reserve of time which they can spend helping others with tough problems.

There are two problems which stand out before all others which I have experienced with people adjusting to this methodology. The first is that people have a hard time understanding the concept that a deadline is a deadline. It is not an "I'm almost there" or an "I think

I can complete it by Monday." I would rather have someone finished a day early and looking for work than to finish five minutes late. The second is getting people to seek help from their peers. The team meetings help people adjust to this concept as it is a relatively non-threatening way to give and receive advice. However, a key to the success of this methodology also hinges on people going to each other throughout the day not only for design and approach suggestions, but for actual coding and debugging help as well. A giver of help today can be a receiver of help tomorrow, but traditionally, people get caught up in the idea that giving and receiving help creates a heirarchy or pecking order, and people are resistant to seek help when they need it, or wait to seek it until they have wasted vast amounts of time wrestling with a problem.

At the team meeting each member reports on the status of the work they have committed to complete at the previous weeks meeting. If it becomes apparent that, for any reason, a member of the team is falling behind or having difficulty completing their commitments, this is a reason for concern and action. The action may be to rearrange assignments, possibly offloading some of that person's commitment. The concern should be for the underlying cause and the effect it has on the team morale. The rearranging of assignments and responsibilities is essential to maintaining the kind of fluid, goal oriented energy that the must continue throughout the project. For every team member to know that the goal is the teams success, and therefore the project's success, puts a burden on them to not let the team down but at the same time guarantees them that the team will not let them down and that they need only reach out to obtain the kind of support they need, without penalty. Every once in a while, a team member will come along who wants to work the system. They never want to do any work and are only concerned with their own comfort and convenience. The team quickly recognizes this type and naturally resents them. You really have two choices, you can get them off the project, or if that's impossible, you can pigeon hole them with useless assignments that no one cares about and don't really need to be done.

At the team walkthrus and meetings, all team members are encouraged to participate equally in designing and critiquing the system as it evolves. This gives each team member a real sense of ownership of what is evolving, while at the same time preparing them to take

over work in any area if or when it becomes necessary. It also results in a far better system because many different perspectives and backgrounds come together to suggest the best possible solution.

The weekly team meetings are essential. Other project managers who I have trained in this methodology have sometimes thought they could short cut this by meeting with the individual team members, or smaller subsets of team members, to avoid the high cost of such meetings. It never works, and none of the projects managed that way have come out "successfully" by my measurement of success - on time and on budget and meeting the user requirements.
Bypassing these meetings destroys the team dynamic, and in my opinion and experience also destroys the project's ability to be successful.

User demos are an essential way of securing the users' participation as team members. Demoing to users one on one is always helpful and necessary. Demoing to users in a group gets the users to communicate and agree as a group, an almost impossible task otherwise, even within the same department. A user from each major department even peripherally affected by the system should be at these demos, but at the same time the number should be kept manageable - ideally six and never more than ten. A user can either share with the group or keep internalized whatever reaction he or she is having to the direction the project is taking with design, training and implementation. If it is shared, it helps the group focus on issues and resolve them. If is is internalized, the user must later explain to the group why their issues weren't brought up earlier, if they dare.

One major issue is whether developers should be in the demos or not. I favor having a user liaison or "project ambassador" who knows the system inside and out demo the system and give feedback to the developers. This may be the training person discussed below, or it may be one of the developers. If it is a developer it must be someone who has a very special temperament. The problem with having developers in demos is they start explaining why they designed things a certain way and it usually ends up evolving into them arguing for their design while the user argues for what they want. These demos are for listening to the user reactions and issues, not arguing solutions, unless the users want to argue among themselves. The advantage of having a user liaison who is a developer, is that this person can also have the

role of resolving design issues which come up. It tends to be counter productive to have each developer handling their own design "loose ends" with the users. This is because of a variety of factors. Not all developers communicate well with users. Many times two or more developers will have unresolved issues which overlap, where the considerations for one affect the others. Finally, it is far more time and cost efficient to have one person co-ordinating and handling this area.

QC, Documentation and Training

Quality control, user documentation and training are best handled by non-technical, and preferably non-MIS personnel. I have evolved a system whereby I work with a documentation and training specialist who used to be a teacher. I involve her in the project after the first major user demo, and she is involved thereafter until the system is implemented. She merges all three activities.

The QC/documentation/training person(s) learns the application from the user perspective by working with the users to evolve a training plan which will be oriented to their actual daily routine and problems. She learns the system through demos and questions, but mostly through exploring it on her own, trying things, finding out how friendly the system is or isn't, where it is and isn't foolproof, and where things work intuitively and where they don't. This entire learning process on her part provides the developers with a level of testing, debugging, quality control, and user friendliness "tuning" that they would otherwise probably never be able to duplicate.

The biggest payoff of the entire process, is that the perspective she brings is that of a user not a programmer, or developer, or MIS person. The way she moves through the system is the way a user would move through the system, the things she does are things users would do. What this enables us to do is correct problem areas before users ever see them. This gives the entire system far more credibility and acceptance with the user community.

When the system is "ready," it is debugged and the final demo has been done, our trainer holds a training preview class. This class is a test drive of the training process with a very select classroom of

handpicked users. This is a real production preview for the development team, as this group finds all remaining issues that our documentation and training person missed. It is best to do all training with part or all of the converted data so that the integrity of the conversion is tested, as well as lending credibility to the test environment and examples. We leave a window of time between this "preview" class and the first real training class to fix all these remaining problems. When training commences the system is clean, user friendly, and functional. Users do not get discouraged and frustrated by encountering problems during training and early use.

When this process breaks down it is usually because one of two things occurs. First, and most common, the users will insist that only management level people attend demos and the training preview classes. What results is that the system that the actual users of the system first see is frequently not what they need, they have no ownership of it or in it, and they frequently come out of training rejecting the system either literally or psychologically. The second is that the users are "too busy" to participate as they need to in demos and training. Their management has not seen to it that they are committed members of the team as well. This results in delays, bugs and failures in meeting requirements which are always blamed on MIS.

When the process is supported, the system which is implemented is amazingly clean and bug free, and inevitably is a very close fit with the users original "dream."

Understanding a bit of how the development and implementation process works, let us now look at how a plan is developed which will provide the support structure for the development.

The Plan

Once you have the estimate, you can make the plan. We are going to assume that the estimating process is complete, because once we have discussed the plan, it will be easier to understand how the estimate is used, and thus how it is put together. The plan outlines milestones, dates when things will happen, what will happen, and how many of the project hours will be used to get there. Milestones should never be more than two to four weeks apart, depending on the total scope of the

project. I usually like to have user demos be my milestone activities, culminating in training and implementation dates.

The plan is created by deciding what the implementation date is and working backwards. Everything, of course depends on the overall size of the project. Leaving a month at the end for training and implementation makes sense for a 10,000 hour project but not for a 1000 hour project. Similarly, letting several weeks elapse between user demos (formal - the informal ones are going on all the time) makes sense in a 10,000 hour project but it would be a disaster to go more than a couple of weeks in a 1000 hour project. For each user walkthru, there is a corresponding developer deadline and project team walkthru. This means that all work needed to be completed before the user walkthru must be complete (developer deadline) and that the entire project team will participate in a "dress rehearsal" of the real user walkthru (project team walkthru). Depending on the amount of material deliverable at each demo, the developer deadline and project team walkthru date should be two to six working days before the user demo, to allow final debugging, changes and polishing identified at the preceeding project team walkthru. Once you have developed the plan, it may become obvious that you are going to either have to add more people to the project or extend the implementation date.

Now let's discuss the mechanics of developing the plan. We will use the little customer maintenance project estimated below.
The system is due July 1st. I can't possibly get approval and get started before May 15th. I have to leave the week before July 1st for training and implementation. That gives me about 6 weeks before July 1st and 1 week after (post implementation support of problems which is really spread over more than one week but is only part time). So seven weeks to get 316 hours of work completed. I figure each person on average can't be estimated at more than 30 hours per week. So this works out to about exactly one and one-half people over 7 weeks to complete. That's about perfect, because I had planned to have one full time developer and one half time project manager/DBA/developer. So, in this case I don't have to add or take away people from the project, or move the delivery date. My plan will look something like this:

5/29 - Team Walkthru of 6/2 demo

6/2 - First User demo - Review preliminary prototype of all on_line deliverables

6/9 - Team Walkthru of 6/11 demo
6/11 - Second User demo - Review fully functional prototype of all on-line deliverables, review preliminary report output
6/17 - Team Walkthru of 6/19 demo
6/19 - Final User demo - Review of "finished" system
6/23 - Team Walkthru of final changes requested 6/19 and prep for training
6/25 - Training commences
7/1 - System in production (There should still be at least 50 hours left for support at this point)

This project is too short and too small to track the number of hours I expect to use to get to each demo. In a large system, however, I would definitely want that information.

The crucial point here is that dates never slip. Once the project plan is established it is published for users, management and team participants. Slipping dates is not an option.

Now that we have looked at the development process, and the plan which supports it, it should be easier to understand how we do the estimating.

The Estimate

The foundation of any successful project is a good up front estimate, and I find it impossible to successfully manage a project without one. With a 4GL project, however, there is a significant effort which must precede the estimating process.

A requirements analysis phase must be done which at a minimum includes a rudimentary entity analysis. The results of the requirements analysis help to define the scope of the project. The primary maintenance and inquiry areas, with as much substructuring as possible are then laid out. For example, Customer Maintenance is a primary maintenance/inquiry area, supported by substructures such as ship to information, bill-to information, contact information, etc.. This is followed by a similar process of enumerating batch processing which will be needed by the system, such as G/L interface, and reporting, both operational and management. When all on-line processes, batch processes and reporting have been identified to the best of ones ability, estimates can be assigned to each entry or

"task." Reporting is the most difficult area to estimate in a new system. If this is a brand new system, chances are that screens and "operational reporting" such as invoices, bills of lading and so on, are pretty predictable based on information and work flow. However, reports are rarely as easy to define, particularly because it is difficult for users who have never been "on-line" to envision what information (and in what format) they will actually look up on-line versus what information they will want to see in report form. If a system is being replaced, users typically insist they need all the reports they have now plus some, and in the exact same or slightly improved format, despite the fact that they don't use or look at half the reports, the new on-line inquiries will reduce their need for another 25% and redesign would greatly benefit the remainder. So what usually happens is that you develop the reports at least twice. I only mention this because everyone tends to think of reports as the "easy" part, but they typically take up a disproportionate amount of the development and review time.

The first place where estimating typically breaks down are estimates based on what someone thinks it will take to code the screen, process or report in question. I triple that number. The first third of my estimate is indeed how long it will take to code the task at hand. The second third is how long it will take to keep changing it until it works perfectly and everyone agrees it's done. The final third is reserved for all the changes which will be made and time which will be spent after the users start actually using the delivered product, in training and production, a cost which is almost never included in estimating, but which is almost always considered by management to be part of the cost to complete the project.

The next place where estimating usually breaks down is in two areas which are rarely included in estimates. First is the cost of data base administration. Whether you have a dedicated DBA, or whether everyone does some of it, it is a high overhead item. Usually, you will want to tack on a load factor of at least 10% for this task. Second, and most important, is the cost of project management, overhead, meetings, etc. Include a load factor of 20% for this task.

Your final estimate should look something like this:

Customer Maintenance Screen	40 hours
Ship to Maintenance	20 hours
Bill to Maintenance	20 hours
Contact Maintenance	20 hours
Table Maintenance	40 hours
Customer Archiving Process	40 hours
Customer Alpha List	30 hours
Conversion	20 hours
Subtotal	230 hours
DBA Overhead	23 hours
Training	12 hours
Project Management Overhead	51 hours
Project Total	316 hours

It is imperative that your estimate be broken down into "task" level items. Otherwise, managing the project will break down over time.

You will notice that there is no design time built into this estimate. There is a reason for this. In very large projects, or projects which lack sufficient information or definition to begin setting up a prototype once the requirements analysis phase is complete, it is best to plan a design phase which has its own estimate. In a very large project, I will typically try to reserve a two to three month window at the beginning to interview users, brainstorm design, and begin to set up the dictionary data structures. This is usually best done with two people (or more if absolutely necessary), preferably two people who complement each other rather than two people who think similarly. Once the design phase is complete, it is appropriate to redo the original estimate and task list to reflect new thinking about what the system will really look like. A word of caution here. I cannot stress enough how important it is to be perfectly clear with upper management that your initial estimate is just that - preliminary - if you plan to do a design phase and thus re-estimate after completing the design phase. Nine times out of ten, upper management will conveniently forget that there was to be re-estimating following design and will try to beat you into

living with your original estimate. This is a setup for failure. In smaller or well defined projects, there is sufficient "design" time built into the estimates for a prototyping methodology, that a separate estimate for design, followed by re-estimating the development, is not typically needed.

Conclusion

The ability to make very rapid changes and corrections, even to basic design, in 4GL development environments, both eases and complicates project management using 4GL technology. One must start with a solid estimate that closely reflects the scope of the project. An overall plan of deliverables with demo dates attached for the entire length of the system must be established up front and adhered to, even if there are occasionally items which are not ready. The development team must be trained to work as a team, and this must be reinforced by regular weekly team meetings where the entire team has the opportunity to observe and participate in the evolution of the system as a whole instead of isolated pieces. They must look upon one another as peers and equals, helpmates whose successes are shared by all and whose failures are also shared by all, not singling out stars and slugs. There must be a plan in place which allows for intensive testing and shakedown by a user oriented person, so that this is not left to the devices of even the friendliest MIS person or users once the system is implemented. When this methodology is embraced, truly phenomenal results are possible with virtually any team.

Paper Number 7011:

95LX Tips, Tricks and Talents

John L. Vandegrift
Hewlett-Packard, Company
2015 South Park Place
Atlanta, Georgia 30339
(404) 850-2322

This paper is an introduction to the HP 95LX, the "palmtop". This paper is meant to be an introduction to the palmtop. Anyone new to the 95LX will find this paper a good survey of the capabilities present in the palmtop. The features of the 95LX will be covered, as well as some of the hardware and software accessories that are available to make the 95LX more flexible. Additionally, there will be tips along the way to take advantage of the capabilities.

Why the palmtop? When it was introduced two years ago, it defined a new area of computing and was an immediate success. As soon as I heard about it, I couldn't wait to get my hands on one. I checked the local computer stores on a weekly basis, and bought one of the first units available to the general public. Prior to buying the palmtop, I had been looking at a number of ways to automate my daily tasks. I kept an organizer that had a calendar, a phone list, a to-do list, and other typical organizer features. I also had a calculator that I kept in my briefcase and a notepad that I carried in my shirt pocket for occasions when I wasn't near my organizer. Basically I wanted an automated solution that would allow me to replace the organizer, the calculator and the pocket notepad that I was currently using. There were electronic pocket organizers out there already, but they were proprietary and limited in functionality. There were also portable computers available, which provided great functionality, but were large and inconvenient. I wanted a cross between the electronic organizers and the portable computers, and HP answered that need with the palmtop. And, just as it was with the public, it was an immediate success with me.

The palmtop is every bit an MS-DOS based computer as the personal computer is on your desk. It is simply built in a small package that runs on two AA batteries. By today's processing standards, it is slow. But it is not the processing power that sets this computer apart. It is the ability to take an MS-DOS based computer with you wherever you go. The palmtop quickly becomes a trusted ally without which you would be lost. My palmtop has gone just about everywhere I have gone, including up a few trees. It is my constant companion at work and in meetings, as well as business trips. I have taken it on hikes in the woods and up in trees on hunting trips. And it is rugged.

Let's take a look at the palmtop. It has a calculator like keyboard, laid out in the familiar QWERTY fashion. It is too small to get all fingers into the "home" positon for touch typing, but the familiar QWERTY arrangement of the keys makes it rather simple to become effective at typing with each index finger. Some people like to hold each end of the palmtop in their hands and type in a similar fashion with their thumbs. You know you are in a group of 95LX enthusiasts when an animated discussion breaks out about the best way to type on the palmtop! While I would not want to type in a novel on the palmtop, I have done a lot of typing on it, including rough drafts of papers I was working on.

The screen of the palmtop is an LCD display, with 40 columns across and 16 lines down the screen. It also has the ability, under most circumstances, to scroll from side to side and up and down to read screens of programs that are meant to be read on the standard 80 by 24 screens of the larger computers. The screen is not backlit, but is very readable.

There are several applications, and MS-DOS version 3.22, built into the internal 1 megabyte of ROM. The applications that are built into the palmtop are called Personal Information Managers, or PIMs, for short. These PIMs provide a lot of the basic usefulness of the palmtop. Add the capability provided by being an MS-DOS machine, and you have a powerful computer. Let me give an example to illustrate the point. I noticed that the calculator PIM was missing the ability to do base conversions. I was a little disappointed that the 95LX did not provide for this functionality. I logged into CompuServe and checked out the HP forum where the 95LX information is kept. Sure enough, there were already 3 programs written for the 95LX that added this capability. I downloaded all 3 programs on my PC at home, selected the one that I liked, and transferred it to my palmtop. So the MS-DOS capability allows anyone used to developing tools on a PC to develop palmtop utilities. And a lot of software has been developed for the palmtop. There are many programs to choose from that are free, shareware, or marketed packages.

Now that we have seen some of the advantage of having an MS-DOS based computer, we'll examine the PIMs that are built into the 95LX. First, the palmtop works in an "instant-ignition" mode. This means that you do not have to boot the palmtop every time you want to use it. It turns on and off, but leaves the applications running where they were. Note that I used the plural here. You can have more than one application open and running at a time, although the palmtop only allows one to execute at a time. So you can be in the middle of a Lotus spreadsheet and instantly hop over to your calendar to make an appointment. When you are done, you can return to Lotus, exactly where you left off. No waiting!

The first PIM that I will describe is the Filer. The Filer provides the ability to do file management on the palmtop. This eliminates the need to drop down to MS-DOS to execute the necessary commands. Instead you can push the key that represents the Filer and you are immediately put into the Filer application. From within the Filer you can list, copy, move, delete, view and rename files as well as execute programs, print files, create new directories and perform file operations on entire directories. Additionally, if you have the Connectivity Pack, which I'll mention in additional accessories you can get for the palmtop, you can use the Filer along with the PC Filer from the Connectivity Pack for transferring files between the palmtop and your PC. One of the nice features of the Filer is that you can look at two different views of files at the same time. Both of these can be local to the palmtop that you are using, or one can be a remote system that you are transferring files with. The filer has the ability to use its built in RS-232 port for remote transfers, or it can use the built in infrared port with another palmtop. So if you have a friend that has a palmtop, you can share files without having to hook the two palmtops together, using the technology present in the infrared port, which works in a fashion similar to many remote controls for television sets. In fact, there are some programs available that will allow the palmtop to control your various electronic devices, such as a television or stereo, just as your remote controls do. I find the Filer a convenient application for quickly manipulating files on the palmtop, and rarely drop down to the MS-DOS prompt for dealing with the files on the palmtop. It is also from the Filer that you get to MS-DOS when you need to, although I don't find much of need to get down to the operating system level as the PIMs provide much of the functionality that I need. Note that to get to MS-DOS, all other applications must be closed before you go into the Filer. Close all applications, enter the Filer, press MENU, and the press S for System. This will bring up the familiar MS-DOS prompt.

Next there is a data communications PIM that has a lot of basic capability built into it. It can be used for emulating a VT100 or ANSI terminal. It also has the ability to do some basic mapping of characters while emulating a terminal. The interface can be RS-232 or the infrared port. For file transfers, it has the Kermit and XMODEM protocols built into it. It will also do basic text transfers. There are a lot of parameters that can be set, such as parity, baud, duplex and a variety of other characteristics for those that like to get into the finer points of data communications. For those of us that like to "plug and play", there are several default configuration files included with the palmtop, including ones to access CompuServe, Dow Jones and Genie services. Even if these aren't the services that you will be connecting to, loading these profiles and looking at the settings may be a good headstart in getting connected to your own service.

The next PIM you will see on the keyboard is the appointment book. This is a full featured appointment book that allows you to replace whatever type of calendar

you are currently using. By default, you are looking at your calendar a day at a time. There are several options a number of settings available to you in the appointment book, such as displaying the 8 to 5 part of the day whether there are appointments or not, or only displaying the actual appointments for that day. At the core of the appointment book is the appointment. You can schedule an appointment and it will show up as a one line entry in the appointment book. You can set the start and stop time for the appointment, making the start and stop time the same if you are not sure when the appointment stops, or it is an event that doesn't really have a time, such as noting a holiday. You can also set alarms for any of your appointments. So you have the ability to enable or disable the alarm for a given appointment, as well as set the lead time for the alarm. There are a number of defaults that can be set for the appointment book, such as what the default lead time or default appointment length should be. This is handy for customizing the appointment book to your own lifestyle. You can also set up repeating events, such as a staff meeting that always occurs on the 2nd Wednesday of the month. In addition to looking at the appointment book day by day, you can also take a look a month at a time. This is handy for doing longer range planning. One feature that I miss in the appointment book is that you can't look at the calendar a week at a time. But there is a program available, Week at a Glance, that makes up for this. It can be set up as a hot key so that it is available along side your appointment book. I'll talk more about available software further on in the paper. There are cut and paste features to the appointment book as well as the ability to tack on a note to any of the entries in the appointment book, in case you need to further elaborate on a given appointment. There is also a search capability that allows you to find a string of characters in your appointment book. In addition to these functions, you can carry a To-Do list in your appointment book. It is similar to a To-Do list you might make out, but this one is electronic and time sensitive. To-Do's that have not been done today are automatically carried forward to tomorrow. You can prioritize the To-Do's and make notes on any of the entries in the To-Do list. One other feature that is interesting in the appointment book is a world clock. There is a file of cities from around the world and the times of those cities. You can edit this file and have Daylight Savings taken into account. The world clock function also includes a stopwatch and timer. The format of the appointment book file has been documented and a number of additional utilities have been written that perform a number of tasks on the appointment book, such as translating the appointment book to another calendar package format.

Along with the appointment book, the phone book is another very useful PIM. It allows you to track the various contacts that you do business with. I use the phone book as a somewhat generic database. Not only do I put business contacts in the database, but I also put entries in there for people that have different affiliations, such as a task force that I may be working on at the moment. With the

searching capability of the phone book application, I can quickly find all people that are in that task force. The phone book tracks such things as name and number, but it also has a large text area, which leaves room for the typical address information. Or I might have the directions to company that I am visiting in there. I also put entries in the phone book for such things as log on information. The name of the system goes in the name field and the text field has the specifics for getting onto the system in question. If I get a new insurance agent, I put the agent's name, phone number and pertinent information in the text field. I also make sure that I put "insurance agent" in the text field somewhere so that I can quickly search for "agent" or "insurance" and find the agent again, in case I forget the agent's name. Both the appointment book and the phone book have the ability to track multiple files. So you could have a phone book for business and another that your friend with a 95LX gave you for new prospects. Switching between the two is a few keystrokes away. There are some handy cut and paste type of features in the phone book as well. Like the appointment book, the phone book file format has been documented and there are a number of utilities out for performing a wide range of tasks against the 95LX phone book files.

The next PIM you encounter on the keyboard of the palmtop is the memo PIM. This is a basic word processor, providing the very basic functions needed in creating and maintaining documents. It is not a full featured document processor by any stretch of the imagination. But it is more than enough to get some basic documents entered into the palmtop until you have the chance to get back to the office and upload the document to work on it with a full fledged document processor. The memo application has the basic cut and paste capabilities as well as the ability to search through a file for a certain string of characters. It isn't going to become your favorite editor, but it is fairly intuitive and functional. It deals with files as flat ASCII files.

The button on the palmtop to the right of the memo PIM is labelled "123". And yes, it is a full blown implimentation of Lotus 123, including some basic graphing functionality. The version of Lotus 123 is 2.2, with a few changes made to accomodate the palmtop, such as dealing with the smaller screen. Naturally the size of the window displayed when you start up the Lotus 123 application is set to the 40 by 16 character display of the palmtop as an example. The graphics are also tamed down a bit, since the graphic resolution of the screen is less than most PC computers in use today. All of the Lotus 123 commands and keystrokes are compatible with the palmtop version, which means that any spreadsheets or macros that you create should work on either your PC version of Lotus 123 or the palmtop version. The other difference you will notice right away in using Lotus 123 on the palmtop, aside from the screen size, is that the keyboard is laid out a little differently. Due to the compactness of the keyboard, it takes some searching at first to find the location of the special keys. After you have used the palmtop

for a while, you will be used to finding the various keys you need. But think of it, Lotus 123 available at your fingertips anywhere you go! I won't attempt to tell you of the features of 123, as you are probably familiar with it already, not to mention that a book could be written on just 123 alone - and many have. It is worth mentioning some of the unique aspects of having 123 built into the palmtop. In my opinion, one of the nicest features is the ability to link in results from the calculator application to a 123 spreadsheet. For example, you could set up an amortization schedule for your home mortgage in the calculator, and once you are satisfied with the components of your calculations, you could then have a Lotus 123 spreadsheet automatically created with each payment of the amortization entered in the spreadsheet. Then you could pull up the Lotus 123 application and further enhance the spreadsheet. So the calculator is linked to Lotus, or can be, for your calculations. One of my favorite applications for Lotus is a spreadsheet that has the format of the Travel Expense Report that we use internally. When I take a trip, I fill out the expense report while I'm on the trip. Quite often I finish the expense report as I am flying back from the trip, so that all I have to do is upload the spreadsheet when I get back to my desk and print it, ready for approval.

The last PIM is the calculator, which I made brief reference to in the Lotus paragraph above. First of all, you get to choose whether you like the Reverse Polish Notation, RPN, or the more standard algebraic notation. Your choice! Next, the calculator includes a number of powerful functions, like most of the calculators that you can buy with HP's name on them. There is a conversion section that seems to be very complete, allowing you to convert currency, length, area, volume and mass. There is a fairly complete selection of math functions, such as logarithms, absolute value, factorials, PI of course, trig functions, etc. There is a nice section in the calculator dealing with the Time Value of Money, or TVM. Here is where you might want to compute those new house payments for a refinanced mortgage. Then turn it into an amortization schedule, load it into a Lotus spreadsheet and perform some more calculations in Lotus 123. There is also a very powerful equation solver built into the calculator application. This allows you to write your own equations. These equations become function keys, custom tailored to you business, for example. You define the equations using variable names, enter the known information and then solve for the unknown. The Solver can receive its information from a Lotus 123 spreadsheet as well, and you can graph the results.

That takes care of a quick introduction to the PIMs on the 95LX. In addition, the early version of the palmtop had 512k of RAM, which more recent models having a full 1Mbyte. This RAM can be divided between system memory and a RAM disk. The RAM disk is drive C:\. You can control how much of the memory is system and how much is devoted to drove C:\ in a setup menu. Additionally, the palmtop makes use of a slot that is PCMCIA 1.0 compliant. In this slot you can

put RAM or ROM cards. It is logically accesses as drive A:\. For example you might buy a ROM card that has a thesaurus/dictionary on it. I have a RAM card in mine that is 1Mbyte in size. The RAM card can be used to back up the contents of drive C:\.

There are a lot of accessories available for the palmtop. And most of these are available through your local computer dealer, or through magazines that carry information about the palmtop, such as the HP Palmtop Paper. Third party vendors supply such items as "docking stations" which expand the connectivity of the palmtop. There are cradles and docking stations that will link your 95LX into faxes, modems and pagers. One pager/cradle solution for the palmtop will allow the 95LX to become an alpha numeric receiver of pages, including editing and cataloging capability for the messages. And the cards that are available for the PCMCIA slot are reaching 10Megabytes and beyond.

One of the most popular add on applications for the 95LX is a program called 95Buddy. Basically, 95Buddy allows you to do more with the palmtop with far fewer keystrokes. It is a shareware program and you will find it installed on a lot of the palmtops out there today. For example, CTRL-B will automatically generate a timestamp in memo, phone or appointment book applications. You can also do such things as rename the fields of the phone application. 95Buddy is basically a short-cut customizer for the 95LX.

Another application out there for the 95LX is a program called SWITCH. It allows a host of programs to work fully in conjunction with the built in applications. One feature of SWITCH is the ability to run DOS programs without exiting built-in applications.

And there are a number of disk stackers that can take your RAM card and double the capacity of it. There are some that are unique to the palmtop, and there are other disk doublers that are used on the palmtop as they would be used on a regular PC. A word of caution. Just because a disk doubler works on the PC, don't assume that it works on the palmtop without checking. There are differences between the electronic disks on the palmtop and the "regular" disks of PCs.

Of course there are a whole rash of games available for the palmtop. In fact, there are two games that are built into the palmtop. Hearts and Bones is one and TigerFox is the other. There are chess, backgammon, Tetris, Blackjack, Adventure, poker and most of the favorite games available.

There is a lot of development going on in the wireless technology for the palmtop. For example there are two-way wireless data communications being developed. One company has technology that will provide a mobile data package

capable of supporting mobile computer users with wireless E-mail and remote access to on-line databases and spreadsheets. And the new developments just keep coming.

One of the best sources of on-going informaton for the palmtop is the HP Palmtop Paper. It is devoted to the palmtop and comes out every other month. It has trouble-shooting tips, reviews, news from HP, tips on using the palmtop, etc. It is full of information and just reading the ads is interesting to see what is new for the palmtop. There is a reference to this magazine in the bibliography. CompuServe has a couple of HP forums on it and is a good source of information from HP, as well as users like yourself. It is also a good place to pick up the latest versions of a lot of the programs that are free or shareware. Many of the programs I have picked up are available on the CompuServe service.

As I was writing this paper, HP introduced a new palmtop, the 100LX. It looks a lot like the 95LX, with some major differences. The PIMs are similar, but there is now a database engine that drives the phone book and appointment book applications. You can also create your own database with the database engine. The screen is now a full 80 by 24, although the characters get rather tiny at this resolution. But they are very crisp. If this is too small for you, the 100LX has the capability to change the screen resolution for many of the applications. The case itself is just like the 95LX. And the slot, drive A:\, is now PCMCIA 2.0 compatible.

Bibliography of Readings

The following list includes a number of sources for learning more about the HP 95LX palmtop and were sources that I used in writing this paper. For those interested in finding out more about the 95LX, the books listed should help you.

Monday, Lori and Tracy Ann Robinson. (1991) "Using Your 95LX".

Goldstein, Hal. Editor. "The HP Palmtop Paper", (515) 472-6330.

Developing HP 3000 Software Using a Unix Workstation

by Ralph Carpenter

INTRODUCTION

There are some questions the reader might be considering:
* "How can I become more productive in the development of HP 3000 software?"
* "Is there some combination of hardware and software anywhe on market today that packages the best "bang for the buck" for an HP 3000 development team's use?"

These are questions that many managers of HP 3000 software development teams and members of those teams struggle with in the ever changing world of competition for our budget dollars; and these are the issues addressed by this paper, in which I discuss my experience with the HP 9000 Unix workstation in the context of developing HP 3000 software.

I am a veteran Hewlett-Packard employee, and my current assignment is the support of the MPE-V operating system with the Software Technology Division's R&D organization in Roseville, CA. During my 20 years at HP, I have worked with a variety of development platforms and environments, including:
1) ASCII terminal hardwire linked to an HP 3000,
2) LAN connected PC,
3) modem connected home PC, and
4) LAN connected HP 9000 Unix workstation, the subject of this paper.

DEVELOPING MPE SOFTWARE IN A UNIX ENVIRONMENT

The evolution of the development environment that my colleagues and I use today began two and a half years ago, when our team was using LAN connected PCs for our development work in Roseville. Our team's primary assignment at that time was developing PC software, and only occasionally was one of us asked to work on HP 3000 software. However, on those occasions, we found that there was a dramatic difference in our productivity between the two development environments.

For PC development, the Microsoft Software Development Environment -- SDE -- which is based on the CodeView development/debugging environment, provides the PC software developer with a view into his/her code at all times during the development cycle; the MS-DOS operating system is kept out of sight during the entire CodeView session.

On the other hand, the HP 3000 -- as viewed through a terminal emulator -- forces the PC based HP 3000 software developer to leave the context of one HP 3000 development tool in order to enter another tool, and in general the HP 3000 tools appeared antiquated in comparison with the Microsoft SDE. It was clear from this experience that the PC platform was less than satisfactory for use as a primary interface for developing HP 3000 software.

For the past two years, I have worked with a LAN connected HP 9000 Unix workstation. This selection of equipment was provided for my use during a recent 7-month assignment at the HP Pinewood site in England. While there, my assignment was to develop Unix software, and so I became familiar with the Unix development environment.

Later, upon my return to California, I was able to convince my management of the productivity gains afforded by the Unix workstation platform, and so a Unix workstation was provided for my use there as well. Again, I was asked to focus on developing Unix software, then underway in Roseville.

Recently I joined the SWT R&D team in Roseville, which is supporting the MPE-V operating system and related products. To this assignment, I have brought my Unix background and training to bear, and today I serve as a Unix mentor to my colleagues in addition to my normal software development duties.

Let's take a look at the current situation with the MPE-V development team. First, our team members are collectively more productive and effective: We generate more lines of higher quality HP 3000 code than we had previously generated using LAN connected PCs. Our MPE-V customers view the team as being more successful and better organized to meet their needs. Secondly, our management views the team as being better organized, demonstrating better focus on their customers' needs. We are in charge of our time, and our tasks are better organized, because we can now visualize all of our tasks -- both pending and active -- at a single glance.

Now let's look back at what it was like before the development team had their LAN connected HP 9000 Unix workstations. The team was relatively unproductive due to the limitations of the LAN connected PC environment. Our customers, both internal and external, viewed the team as being less than fully successful in meeting their needs. As individuals, the team members were in general unorganized with time commitments and with tasks, and this situation led to poor project delivery from the team working together as a unit. Our managers didn't view our performance as being very organized and couldn't understand the relationship between the team's priorities and customer needs.

What happened to effect the top-down change that was needed? First, the LAN connected PC development environment was investigated, and several limitations were uncovered:

1) A limited view into a single HP 3000 development tool's environment at a time,
2) The MS Windows screen appeared cluttered and yielded relatively poor performance.
3) The MS-DOS 5 DOSSHELL was relatively clumsy for task switching, and forced the developer into a single whole-screen window view at a time.
4) And we found the cost of a high speed PC today not significantly less expensive than the cost of a Unix workstation.

Next the LAN connected Unix workstation environment was investigated, and several key advantages stood out:

1) The Unix workstation provides concurrent views into many HP 3000 development tool environments at any given time by utilizing the technologies inherent in X-Windows, in Motif and in HP VUE (please refer to the following discussion of Unix applications for clarification of these terms).
2) If we were to choose a Unix workstation platform, we would be well positioned to take advantage of X-Windows support by MPE-iX, as well as by HP-UX, should the need arise to develop internal tools for use on either of those platforms.
3) A Unix workstation user can view all tasks at any time, including arriving electronic memos and memos left partially completed, pending appointments, open online reference documents, personal filing cabinet, source editing sessions, debugging sessions, dump analysis sessions, appointment reminders, etc. The visibility of all tasks at a glance improves time/task coordination,and avoids missed commitments.

ROLE OF THE WORLDWIDE CORPORATE NETWORK

A key element in the Hewlett-Packard software development and support environment is the worldwide corporate network. Without such a network,several communications based tasks would be made extremely difficult, if not impossible: Communicating reported problems to SWT for resolution; following-up on reported problems from the development team back to the Response Center and field support teams; coordination among development teams responsible for related HP products; and dissemination of problem resolutions and product enhancements back to the Response Center and field support teams. HP operates one of largest private corporate networks in the world.

Local LANs interconnect all HP professional workers' PCs and Unix workstations. These LANs are located worldwide at HP sites, and the site LANs are interconnected via the X.25 packet-switched public data network. At each HP site, StarLAN, ThinLAN, and ThickLAN (or backbone) is used to link nodes together. An employee places a modem call from his/her home into a Gandalf switch using the employee's PC and signs on; the switch then dials back to the employee's pre-recorded home phone number, thus ensuring a secure connection.

APPLICATIONS

There are many Unix and MPE-V applications available through the use of an HP 9000 workstation. Several are popular with the MPE-V development team: HP VUE, VT3K, SoftPC, HP Source Reader, HP IDAT, MPower/SharedX, Unix Mail -- encapsulated under HP SoftBench, Unix News, PostNote, DateBook, LaserROM, and LAN Manager for Unix. In addition, during leisure time, we use graphical applications, such as Weather and StockChart, and we play graphical games such as Mahjongg, Chess, and Othello.

HP VUE:

HP VUE serves as the primary Unix workstation user interface most popular with the SWT MPE-V team. It is built on top of the industry standard X-Windows, and HP's recent user interface submission to OSF, Motif. HP VUE supports one or more "environments", any one of which can be activated a time. Selection among multiple visual context environments allows developer to customize the screen to suit his/her work requirements. For example, my custom context environments are labelled Unix Mail, News, Administration, LaserROM, MPE-V, and MPE-iX. The availability of a selection among a custom set of environments reduces the apparent clutter arising in a standard X-Windows single environment screen. HP VUE permits any one application to be visible in one or more environments concurrently. For example, DateBook and personal filing can be made common to all contexts, and so be always visible, regardless of the selected environment. HP VUE's personal filing facility retains for each user a window displaying a list of files and directories at a selectable hierarchical level; the files are represented by icons that identify the nature of the file (an executable file icon looks different from a text file icon). By double-clicking on a file icon, the file is activated according to file type -- an executable is automatically executed and a text file causes automatic execution of an editor with the text file as the editable object. The personal filing cabinet is an example of the HP NewWave drag-and-drop functionality that is embodied for the Unix product family within HP VUE. Other actions available in HP VUE include drag-and-drop onto a printer or into the wastebasket, and activation of Unix Mail by clicking on the Mail icon. Other features include screen backdrop and color scheme selection, remote host selection, custom application setup, clock display, and processor load display.

VT3K:

A key Unix application is VT3K. VT3K controls the virtual terminal communication interaction from HP Unix to MPE systems, and is the basis for the HP 3000 applications selected by the MPE-V development team.

SoftPC:
SoftPC is a Unix software and hardware product. The hardware component is a printed circuit board containing an Intel 286 processor chip, Intel memory chips, and workstation backplane interface circuitry. The software component controls the PC-on-a-board in executing MS-DOS applications selected by the workstation user, mapping graphics from Intel display memory into that of the workstation.

HP Source Reader:
HP Source Reader is an internal HP software tool that displays MPE-V and MPE-iX source code from a CDROM onto a PC screen. In a Unix workstation environment, Source Reader is used in conjunction with SoftPC, and can thus display in a window instead of on a PC screen. Several such windows can be used to simultaneously display several views of the MPE-V source code at different procedures. In addition different SoftPC windows can simultaneously display different source code -- for example, a procedure in MPE-V and its Native Mode counterpart in MPE-iX. The CDROM drives can either be locally connected to the workstation, or they can be connected to the workstation cluster server, or they can be NFS mounted over the LAN and be connected to a unrelated Unix node.

HP IDAT:
If an MPE-V developer receives a memory dump as part of a customer or internal problem report, he/she can bring up IDAT (for Interactive Dump Analysis Tool) displaying an HP 300 session in a window, using the VT3K terminal emulator. When the developer locates the problem using IDAT, he/she can activate an MPE-V based editor session, such as HPEDIT, and perhaps another window with Source Reader showing the calling procedure, and can efficiently code the change for subsequent testing in yet another VT3K window in which the program under test is debugged.

HP MPower/SharedX:
MPower/SharedX enables synchronized cooperative work between two HP developers, located either at the same HP site, or at distant worldwide sites. The design center for MPower/SharedX is two developers, both equipped with Unix workstations, and an open voice phone line connecting them. When a phone call concerns a mentoring situation, as it often does, MPower/SharedX allows one user to open one of his/her local windows onto the remote user's screen, and to display all of the information in a window onto the other's display. Any modification one of the developers makes to his/her local window is instantaneously updated to the other developer's screen. So verbal communication is dramatically enhanced by visual demonstration of ideas, and active participation by both developers speeds learning.

The MPE-V development team relies heavily upon use of MPower/SharedX and uses it daily in cross-training situations, because an increasing number of products within the MPE-V operating environment continue to be transferred to SWT for ongoing support. Most recently, several HP 3000 networking products have begun to be studied for transfer from the Cupertino site to the Roseville site; MPower/SharedX has played a significant role in follow-up to classroom training sessions and is smoothing the transition, as well as speeding it.

Unix Mail:
The use of Unix Mail is becoming available to more and more HP professionals that so choose, eliminating one of the main organizational barriers to the adoption of the Unix workstation in the HP 3000 developer community. All HP professional workers have electronic mail available. At first, only HPDesk was able to be accommodated on the business side of the organization, and only Unix Mail was able to be accommodated on the technical side of the organization. Now, increasing numbers of HP technical workers with Unix workstations are able to move over to using Unix Mail, because of the advent of OpenMail. OpenMail is a recently produced HP product that links the proprietary HPDesk system using the X.400 industry standard interchange protocol, allowing open communication with Unix Mail, as well as IBM PROFS and other vendors' proprietary electronic mail systems. OpenMail executes on an HP-UX platform.

Unix News:
Unix News is available to the HP developer community through the Unix workstation interface and to other HP workers through the PC executing News with a terminal emulator. News implements the paradigm of an electronic bulletin board. The top level index of the bulletin board is the main topic area. Within a main topic area is the list of base notes, and each base note may have a set of response notes. News main topic areas are of two kinds: Those that are open to the world, and those that are closed to inside-HP (and, of course to inside any other organization that has a News feed). Any News user having access to a main topic area can read a base note in that topic area, can author a new base note, or can read and reply to any base note. E.g. I could start News on the topic of "rec.backcountry"; I could look for a base note containing the subject word, "Mt.Whitney", and I could either browse the available information, or I could make an inquiry of my own by authoring a new base note -- which I would later want to check for responses.

Unix PostNote:
The PostNote Unix application facilitates workgroup coordination, either locally, or between HP sites worldwide. PostNote communication is directed to one or more recipients and is instantaneous -- causing immediate interruption of the selected recipient(s). PostNote allows asynchronous communication by not requiring the actual display of the received message at the time that the note is sent, but queuing-up notes so that they can be viewed at the recipient's leisure -- although, when the recipient wishes to use his/her workstation for any reason, the very first thing he/she must do is to display all pending notes.

DateBook:
The DateBook Unix application is central to the personal time and task management organization for the majority of the development team. With DateBook, past and future appointments can be recorded. Weekly or daily calendar events can be printed, and the month-at-a- glance display indicates whether there are any appointments for a given day -- the current day is highlighted, and days having appointments appear different than do days without appointments. The daily calendar entry is activated by a double-click on the day number within the month display, and depicts the appointment details and permits modification.

HP LaserROM:
LaserROM can eliminate the need for having a private copy of operating system reference manuals. MPE-V, MPE-iX, HP-UX operating system reference manuals are available today on CDROM, as is information useful to the internal HP field organization. The keyword search facility permits instantaneous access to reference manual entries, with keywords highlighted. The Unix system executing the LaserROM application must either have a local CDROM drive, or must have NFS access to a CDROM drive having the appropriate manual set physically mounted.

LAN Manager for Unix:
Within the SWT R&D organization, there are a variety of development platforms in use, including LAN connected HP 9000 workstations and LAN connected PCs. The LAN connected PCs use LAN Manager Client software and access virtual drives on Unix servers through the LAN Manager for Unix (LM/X) Server product. Files stored from a PC onto a virtual LM/X disk drive physically exist as Unix files, and can thus be readily shared by Unix users of the server -- generally, a cluster server. Using LM/X, the development teams can share information in spite of the differences in hardware platforms used. The SWT R&D management team also uses LAN connected PCs, and so they too can communicate and coordinate with the development teams through the use of LM/X.

SUMMARY

Our workgroup of HP 3000 software developers working on MPE-V is much more productive today using their HP 9000 Unix workstations than when we were using LAN connected PCs. It is especially cost-effective in HP's distributed laboratory environment of geographically distant sites, when used in conjunction with applications designed for use with the company's private wide area data network.

Our MPE-V customers are better satisfied with the delivery of needed functionality and with timeliness of defect repair to MPE-V problems, and thus with improved return on their MPE-V software support contract investment.

If the reader is responsible for the productivity of an HP 3000 software development team, or is a member of such a team, and is faced with a choice among hardware and software platforms, then he needs to consider the advantages offered by the Unix workstation alternative.

Interex Paper No: 7013
Title: LAN Migration Strategies
Presenter: Paul Morgan-Witts
Hewlett Packard Company
19447 Pruneridge Avenue
Cupertino, CA 95014, USA
(408) 725 8900

LAN MIGRATION STRATEGIES

Experiences, Issues, Solution

Good morning, Ladies and Gentlemen. My name is Paul Morgan-Witts, I am the product marketing manager for HP DeskManager, based in the UK. I am going to talk about LAN migration strategies - experiences, issues and solutions.

More specifically, I will be looking at why customers are migrating to PC LAN solutions. Focussing especially on electronic messaging, we will spend a few minutes discussing the different architectural approaches taken by manufacturers. We will look at a number of actual customer's experiences to illustrate the problems users are facing when migrating, and what true needs should be from taken into consideration when developing a migration strategy.

My objective is share some of the problems that we see in such a migration, and some of the experiences other customer have had. I will also suggest a strategy which will help you avoid some of the pitfalls.

WHY ARE CUSTOMERS MIGRATING?

There are 3 principal reasons why customers are migrating. Technology, the market and buyer behavior. Let's look at these one at a time, starting with technology.

What we have seen over the last few years is the sophistication of the PC LAN technology consistently and considerably increasing. This has enabled the distribution of processing in both local and wide area networks. We have also seen a dramatic increase in application availability.

Applications previously found on minicomputers or on large mainframe facilities, whether it be a financial package or large transaction processing application, are now available on alternative operating systems - frequently on PCs.

Another obvious technological advance has been in the user

friendliness of many of the LAN/PC-based technologies. One of the most obvious areas is that of LAN Mail, with products such Lotus/cc:Mail and Microsoft Mail, together with a host of other client providers. The ease of use and graphical environment of these user interfaces is a far cry from the character-based terminal interface that we find on mini and mainframe mail systems.

The second area that is forcing people to migrate is the market itself. A lot of the traditional mainframe solutions are no longer being enhanced. For example, the discontinuance of PROFS and the failure of OfficeVision II, recently announced by IBM, is a fact which is obviously is giving a clear message to users that they need to be considering alternative messaging solutions.

Also, low end solutions are readily available. Instead of having to go to MIS for your messaging solution, you can do go down to your local dealer or distributor and pick up a very cost effective, functionally rich, easy to install solution. However, one of the downsides of this is that although the initial purchase cost may be low, when you start factoring in the ongoing support, maintenance, and the so-called 'soft costs' associated with the reliability and security of the technology, then there is a significant hidden cost that is rarely considered by the people that buy these solutions.

Associated with this increase in application availability has been a shift in buyer behavior and power over recent years. Budgets have frequently become decentralized down to functional or departmental heads; the central MIS group have often lost the ability to dictate centrally what technology departmental or functional managers should be purchasing. This has inexorably led to a situation where usability has become more important than manageability, and in many organizations the subsequent increase in IT ownership costs has been significant.

Although the PC LAN provides tremendous potential benefits in terms of ease of use, worrying limitations are emerging is in the areas of security, scalability, and reliability, most notably in medium to large sized networks. To understand the reasons for this, and the limitations inherent in the technology, it is important to spend a few minutes discussing the two basic type of architecture we see in electronic messaging products. These two types are, respectively, file-based architectures and transaction-based architectures.

PC-LAN technology emerged from the concept of a workgroup sharing files on a hard disk. It did not take people long to realize that if a workgroup were sharing files on a disk, this could easily be developed into a simple way of

exchanging electronic messages. So with an attractive user interface and a shared file on the server, LAN-based mail products emerged to provide basic messaging functionality.

This is what we term a file-share architecture, because all of the users on a particular 'post office' (as the LAM mail 'servers' are called) share one file, where all of the messages are kept.

The typical architecture of LAN mail solution is therefore made up of clients connected to "Post-Offices", where the shared file resides. As all the messages are kept in a single file, when any operation takes place - for example, reading a message, or sending a message - that file is locked. It is not locked for long, but as you increase the number of users, the number people trying to access the same file increases, and performance will degrade as file due to file contention. As the performance is dependent upon the accessing of one specific file, this contention occurs irrespective of machine size.

Although the exact number of users depends upon usage/volume of email, typically 50-100 configured users can be supported by one Post-Office.

Compare this to the transaction-based email systems. Here each message is kept in what is called a 'transaction file'. The contention, should it arises, is over accessing a specific message - in real life, this does not become an issue. This architecture dramatically increases the number of mailboxes supported by each server, to a level where it is the power of the machine, not the capabilities of the software, which limits the growth.

Post-Offices store and forward messages for the users on a particular post office; additional software (and hardware) is required if users wish to communicate between one post office and another. The communication between Post-Offices is achieved through dedicated machines, called "external" or "import export". These machines poll each Post Office in turn, using the standard MS-DOS file copy command to "move" messages from one Post Office to another. Obviously, when the post office is being polled, the single file is locked, and therefore most organizations configure this message transport to occur overnight so as to minimize inconvenience.

One External machine can only handle a finite number of Post Offices - again, this depends greatly upon message volume, but a ratio of 1:10 is typical - and so in large networks another 'layer' of servers is required, resulting in a highly inefficient pyramid-style topology. Another obvious effect is that messages can only be moved from one post office to another when the "External" machine polls a

particular Post Office. There is no concept of a Post Office sending a message in a proactive fashion, as in the true client/server model of more powerful mail systems such as HP Desk. For example, if the External machine polls your server once a day, that is how often you will receive messages from other nodes - once a day.

The Post Office, External and each Gateway (e.g. fax, telex, X.400) require dedicated PCs. This means for each 50-100 users there must be a Post Office and an External PC. (Gateway machines are usually centralized and will be configured by usage expectations.)

The effect of this architecture is clearly seen in the following true example.

Company Scenario A

This company is today running a 15,000-user LAN Mail network. Now, today in the world there are not that many companies - probably less than 10 - that are actually operating LAN mail networks of this size. In this case, the network requires over 700 post offices.

This is clearly an extreme - though accurate - example. The classical PC-LAN pyramid-form hierarchical topology is obvious from this diagram. We have about 15,000 users at the bottom, but then as you go up the network, you can see the MTAs and post offices proliferating. From our discussions with this company, we have been able to identify a number of concerns or issues that they have. Starting off with unacceptable delays. One of the problems that this customer has been experiencing is the delay in sending messages from a user at the bottom of the network to one that is remote. The message has to go up the hierarchy and there have been situations where the messages have been taking almost two weeks to get from user A to user B. This is caused by the PC Mail MTA polling around the post offices at each level of the hierarchy, so causing a problem with delays. Another problem is that you can't track lost messages, there are no tools to be able to track where messages are and there is no way of recovering from failures in the network. A number of times, messages, or parts of messages are being lost.

Company Scenario B

Security is a key issue in certain environments, and less important in others. Unfortunately, the file-based architecture of the PC LAN mail systems opens significant security loopholes that are difficult to address. We've seen a number of companies that have experienced a lack of security with LAN-based solutions. Specifically, in this extreme example, the entire message store was deleted by a user.

In the PC LAN server, as I have described, all messages are stored in one file on a shared file server. This means that everybody has at a minimum read and write access to this file; and because everybody's messages are on this file server, it becomes very easy to corrupt or even delete. Although the contents of the message store are encrypted, it is easy to copy the entire message store, take it home, and attempt to break the encryption at your leisure. It is a fundamentally insecure architecture.

Company Scenario C

Another problem that has been experienced by a number of people who are using LAN Mail systems is loss of messages. One of the problems is that there is no way to recover if the post office/MTA link fails. This means that if the LAN goes down, messages could be destroyed or lost. Currently, tools to recover 'lost' messages are weak in PC LAN environments.

PC-LAN based topologies demand multiple servers: As with all systems, the greater the number of components, the higher the likelihood of a failure. For example, if we go back to the company A scenario where we saw all those levels of hierarchy, the message has got to go through many steps to get to its destination. In doing that, there are many potential points of failure.

The net result of these factors is an increase in the overall cost of ownership of the messaging solution. In 1992, we employed the Gartner Group, a well known independent consultancy organization, to attempt to quantify the cost of ownership of an electronic messaging solution. This is a summary of their findings - I should mention that the actual figures are not important (as they will invariable vary from company to company), but of more importance is the methodology. Detailed reports are available from your HP representative.

Gartner Group broke down the cost of owning an electronic messaging solution into four separate areas. Hardware, Software, Support, and what they termed system integrity.

In all cases, costs were based upon 'street prices', and where the cost was shared amongst a number of applications - for example, the cost of the PC - an appropriate proportion of the total cost was allocated to electronic mail.

Gartner Group concluded that the largest delta in costs was in the System Integrity. This figure was based upon a number

of factors: How reliable is the platform (both hardware and software); how often does it crash, and more importantly, how much time is taken to recover from a system crash. They did not try and calculate the value of a lost message, as that figure would be overly subjective - after all, the message could be a luncheon invitation, or a multi-million dollar order.

The conclusion was that the PC LAN based mail systems were most effective in networks of up to around 40 users. Beyond that, there were no economies of scale. By comparison, the average cost per user of the other benchmarked solutions fell rapidly, only flattening out at the 10,000 user mark.

In summary, the fundamental challenge illustrated by these examples is that if you migrate to PC LAN-based mail solution today, you risk having to trade off manageability for usability.

By manageability I mean the types of features or functionalities which we are relatively familiar with: Scalability, security, reliability. On the LAN Mail side we see the tremendous strength in usability, which is ease-of-use, user functionality, and minimal training requirement. What is really needed is the best of both worlds. Organizations considering migration must recognize the potential problems they will experience with large LAN Mail networks, and invest in an infrastructure and strategy that fundamentally solves the problems and that really is the best of both. The good news is that through emerging standards (for example such as VIM and MAPI), and a trend apparent in the marketplace to separate server from client functionality, this is going to be possible. But what are the prerequisites for this ideal solution? What could it look like?

We believe that a successful electronic messaging solution needs to be built on three fundamental platforms. They are: Architecture, Organization, and Cost.

Looking at the Architecture first of all, the key considerations are areas where mini and mainframe systems have traditionally proved very strong: Good directory services, scalability, robustness, and with excellent connectivity. After all, the value of any electronic mail system is a function of the number of people it can connect to. We will return to the architecture in more detail later on.

Organizationally, we have found that it is generally more effective for the network to be centrally managed. This provides benefits in both the effectiveness of the service,

and in reducing overall costs. However, messaging is also better accepted by end-users if the end user is allowed to choose the client most conducive to the individual's work environment. For electronic messaging, this means mail client of choice: MS Mail, cc:Mail, or any other client.

Moving on to Cost. Cost of ownership is an area we feel is critically important, but often neglected. Customers need to understand how the overall cost will roll up over a period of years, and be aware of hidden costs which may not be apparent at the initial cost of purchase. The supplier of messaging also needs to be carefully selected; electronic messaging is a fundamental infrastructure for your organization, and therefore you need to ensure that the chosen supplier is sound, robust, and has that credibility and track record. Equally, putting in such an infrastructure, it is important that you at least have the option of one-stop shopping, not only for the product or the solution, but equally for the ongoing support and maintenance as well.

To meet these criteria, it seems clear that neither the PC LAN based systems, nor traditional terminal/host configurations, will succeed. The messaging infrastructure (or 'backbone' as it is often called) needs to be based upon true, transaction-based enterprise messaging servers on hardware and software platforms appropriate to an infrastructure component. It is also clear that this backbone infrastructure - the servers - cannot today be based upon PC and PC-LAN technology - although it appears to be indisputable that the PC will become the dominant user environment.

What is needed is a hybrid solution which combines the strength of a true enterprise messaging server, with the ease of use of the LAN Mail clients. A number of companies are moving in this direction, led by Hewlett-Packard. Let's look at what this hybrid solution looks like, and what benefits it offers both to the end-users and the system administrators.

At Interex last year - those of you who braved the hurricane may remember it - Hewlett-Packard announced the OpenDesk Initiative and an equivalent strategy on HP's unix-based email system, OpenMail. a key part of this initiative is what we call the 'Clients of Choice' strategy (the first fruits of which are exhibited here today at Interex). This allows users of PC LAN based systems cc:Mail and MS-Mail to connect directly to HP Open DeskManager as clients, in a true client/server model. Hewlett-Packard's messaging servers provide the infrastructural integrity essential for a messaging backbone; through the support of market-leading

user interfaces as clients to the HP backbone, the users can immediately realize the ease of use and attractive user interfaces which they demand.

This provides the benefits of scalable backbone, able to cope with peak usage in the network. The messaging backbone is secure: The message store is protected, preventing unauthorized use or access. Client/server protocols are reliable, robust, and able to protect the integrity of messages in the event of network or system failure.

This software solution is based upon MPE, a highly sophisticated and high availability operating system that will provide a dependable service, which is fundamental if this is to be a key infrastructural component within your organization. Centralized management of the distributed messaging backbone ensures a highly effective solution. Finally, through open and available API's and through conformance of industry standards, the messaging backbone is well-equipped to become the basis for future messaging-enabled applications, not just person-to-person mail, but in the future more sophisticated messaging-enabled applications such as EDI, Routing and Authorization and Workflow.

Turning to the clients, this strategy conceptually splitting the server component from the client component to provide the end-user with a choice of clients on leading platforms, and the ability to select the client most appropriate to the work environment, without losing any client functionality. After all, users do not want to know details of what server they are connecting to, or why; all they want to know is that a message, if sent, will be received at the destination in an efficient and reliable fashion. It is the administrators who need to be concerned with the logistics of that transport.

To summarize this discussion: When migrating to LAN solutions, the risk is that you trade the manageability aspects such as scalability, security, and reliability for usability. Many of those with LAN mail solutions today will recognize the early symptoms of the problems I have illustrated: These problems must be faced early, as they will only grow and become worse.

There is a solution, that allows them to essentially have the best of both worlds. Through combining an industrial strength messaging backbone that has the characteristics of reliability, security and scalability, with the native support of industry-leading LAN mailing clients, companies have both usability at the user level, plus manageability at the MIS level. We believe that what the companies need, and

what they should be considering implementing, is an enterprise information highway, not a departmental cul-de-sac. This solution will provide such a highway.

PAPER #7014

HOW TO'S OF PRESENTATIONS

Kathy Lee Robertson

Manager-CDBG Central Records (Programmer/Analyst)

Virginia Department of Housing & Community Development
Project Management Office
501 North 2nd Street
Richmond, Virginia 23219-1321

(804) 371-7059

Whether to exchange information, to sell an idea or product, to educate or train, presentations are a fact of everyday life. Many of you have wonderful ideas that others would like to hear, but for one reason or another you never take that step to pull it together as a presentation. Others of you must give presentations as part of your work or extracurricular activities, but still feel some discomfort with your level of expertise. After attending this presentation, you will leave with the knowledge that anyone, especially you, can present with confidence and quality. My goal is to encourage those who have never given a presentation to seek out the opportunity to do so, and to increase the comfort level of preparation and delivery of those who must give presentations in the course of their work.

The following is a compilation of the basic rules and tips gathered from a variety of sources. The areas covered include: 1) preparing and presenting your speech, 2) creating and using graphics and handout materials, 3) using the most commonly available types of PC software and hardware, and 4) how to locate the external resources for creating graphics and hardware purchase or rental when you do not have the time, the inclination, or the software and hardware on hand, to do what you want to do.

PREPARING AND PRESENTING YOUR SPEECH

PREPARATION:

1. Know as much as possible about the following:

   a. The THEME OF or REASON FOR the meeting. The choice of your subject and topics should be appropriate for the meeting.

   b. The SUBJECT of your presentation.

   c. The TOPIC(s) on which you will be speaking.

   d. The AUDIENCE. Are they professional or technical peers? What is the knowledge/skill level? What is the level of interest for your topic (summary or detailed)?

   e. The TIME ALLOTTED. This will shape the content and style of your presentation.

   f. The ROOM SET-UP. This will affect your choices for audio-visual aids and types of participation exercises which are appropriate and will indicate whether the audience will be able to comfortably take notes or read your handouts. For example: Are there tables or only chairs? If just chairs, is the floor level or gradient? Are tables rectangular or round? How large is the room? What are your lighting options? Where are the electrical outlets? The list of questions pertaining to the room set-up are endless; if you can't find out what you need to know, prepare as best you can and be ready to adapt to the actual environment.

2. Summarize in one or two paragraphs the reason for doing your presentation, what type of audience you will have or want to reach, the topics you will cover, and the goals you have for the audience as a result of your presentation. This summary will keep you focused as you prepare your speech, and is the primary information you will use in your introductory statement to your audience.

3. Using your topics as the starting point, OUTLINE the main points of your speech. Several drafts may be necessary. The final version should read in a natural progression from one topic to the next.

The greater your satisfaction with your outline the easier it is to do the remaining steps in preparing your speech.

4. Using your outline, fill in the CONTENT of your speech. Some people will find it necessary or more comfortable to include every word they will say, others are comfortable with descriptive phrases which they use as mental prompts for the actual words to be spoken.

5. Do a dry run through of your speech as it stands to get a feel for timing, for the need of audio-visual aids, and for the appropriateness of audience participation exercises.

   a. If this first rehearsal greatly exceeds the time you have been allotted, cut out extraneous information and reduce the content.

   b. When cutting out portions of your speech, keep in mind the reason you are giving this speech and retain those portions with the highest applicability.

   c. Remember to allow for time for your audience to get settled at the beginning, time for questions and answers, and time for any break periods.

   d. If your speech is a long one (an hour or more), plan ahead to 'shift gears' every 20 minutes or so to revitalize audience attention. This is accomplished by inserting a question and answer period, moving to the next topic, reciting a humorous anecdote, relating a real life experience, changing voice modulation and/or body language, switching to a different type of media, and doing audience participation exercises.

6. If you decide upon including audience participation exercises, use the same preparation approach as you used for your speech. In addition:

   a. The room set-up is important when doing audience participation exercises. If you must do the participation exercises regardless of the room set-up, be prepared to adapt the exercises for the best fit (i.e. if you discover there is not enough room for separating the audience into groups for work

sessions, then adapt the exercise for larger groups or for the whole group).

b. Research available publications devoted to giving effective presentations, many include examples of participation exercises that can be modified to fit your needs.

7. If you have determined that audio-visual aids are a necessary part of your presentation, decide upon which type of audio-visuals are best suited to the presentation, and keep in mind the room set-up.

a. FLIP CHARTS.

(1) Flip charts are fine with a small audience, but lose effectiveness with a larger group.

(2) Use of flip charts requires legible handwriting, a marker which has a thick tip, and constant mental reminders to write large enough so that the back row can read it.

(3) Since using flip charts means at least partially if not fully turning your back to your audience, you must be prepared to lose contact with the audience if you stop speaking in order to write onto a flip chart.

b. OVERHEADS.

(1) Overheads are the most popular choice of audio-visual aids and are very effective in promoting audience involvement and retention.

(2) Overheads work with most room set-ups. However, with round tables set up for group exercises it will be uncomfortable for those in the audience with their backs to the front, so you as the speaker must include in your opening statements a request for everyone to take a moment to shift their chairs to give them a view of the front. You do not want a round table set-up if you are giving a presentation requiring the audience to take lots of notes.

c. 35mm SLIDES.

(1) Slides are just as effective as overhead graphics, and are normally used when photos are needed to illustrate the content of your speech.

(2) New technology allows overhead graphics to be processed into 35mm slides as well as superimposed on top of your photo slides, with dramatic results. There is an expense involved, but if your visual aids are unlikely to change in content over an extended period of time, this can be a good option to take.

d. Video.

(1) Use of VCR's and TV monitors is becoming more widely used. If you have only one TV monitor, it will only be effective with a small group. With a medium to large group, multiple monitors would be required.

(2) Newer technology includes the active matrix full color LCD panel, which turns your projection screen into a wide-screen TV. This is very effective if you must use video, or wish to use a mixture of computer generated graphics and video.

e. Interactive Software Programs are another choice as an audio-visual and as a training aid.

8. Decide upon what handouts, if any, you will use, the level of detail to include, and when they should be distributed to the audience.

9. Prepare your audio-visual aids and handouts, proofread carefully, and reserve any equipment you will need for both the day of your presentation and for practice sessions.

10. Practice your speech, working on voice modulation, body language, gestures, and word usage. Be aware of bad speaking habits such as mumbling, 'you know', and 'uh' and get rid of them! Use your audio-visual aids until you are comfortable with them and with the equipment. Practice until you feel relaxed with what you are going to say, how you are going to present it, and feel an acceptable comfort level.

PRESENTING:

1. Have equipment set up and audio-visuals ready; and test for room lighting and focus.

2. When possible try to have the room already set up for the participation exercises (i.e. table placement, flip charts set up with markers at each one, extra pads of paper, pencils).

3. Relax and have fun doing your presentation.

4. Don't lecture to the audience, SHARE your knowledge with them.

5. Maintain awareness during your presentation of audience reaction and be prepared to adjust your presentation style to regain their attention.

6. YOU the speaker are in control. Remember this and use it. You are the moderator for any audience interaction. If the worst happens and two members of the audience begin their own discussion, try regaining control by strolling through the audience until you are between them, and then begin speaking again.

7. Don't fight nervousness or the 'butterflies', accept it, get past it, and use the adrenalin for peak performance.

8. Don't ever <u>completely</u> turn your back to the audience.

9. If you must turn your back to your audience to write onto a flip chart, do not continue your speech while your back is turned, as your audience will likely be unable to hear you.

10. Seldom is there a perfect presentation, don't let yourself get flustered from one mistake, just continue on and RELAX.

11. Maintain eye contact with your audience. Address your remarks to the entire room, left to center to right to center to left...using a smooth eye, head and body motion. Many speakers have a left or right side tendency, ignoring or only allowing themselves darting glances to the other side of the room. If you have this tendency, work on fixing it, for it will alienate your audience and cause their attention to wander, and encourage chatting among the audience feeling ignored.

12. Be prepared to answer questions, and be honest when giving your answers. The audience can always tell when a speaker is bluffing their way through an answer and will be much more impressed by an "I do not have knowledge of that particular area, but I will check into it and get back with you." You, of course, must follow through and actually get back to the person who asked the question.

13. If you feel that audience remarks or questions are leaving the focus of your presentation, it is your responsibility to tactfully refocus the audience to the subject at hand and get back on track.

14. You have introduced your topic, stated your main points, presented the content of your speech, restated in closing the main points and goals and answered any questions. A simple 'Thank you, I have enjoyed this opportunity to make this presentation' is a smooth closing statement that informs the audience they are released from their role of audience.

15. Be brave and ask individuals from your audience for reaction to your performance. If you consider a negative criticism to have value, fix it before your next presentation. Accept all positive criticisms as readily as the negative ones, and give yourself the credit due.

CREATING AND USING GRAPHICS AND HANDOUT MATERIALS

In this section, only graphics for overhead transparencies and handouts will be discussed as they are the most widely used method of audio-visual aid.

OVERHEADS:

1. Use overheads to emphasize the main points on which you are speaking. Do not include everything you will say as the audience will stop listening and responding to you, and will just read the overheads while tuning you out.

2. Make your overheads attractive to look at, i.e. pleasing to the eye, and space apart each point so they are seen as being distinct items.

3. Use concise phrases, not complete sentences.

4. Do not abbreviate, use buzzwords or slang.

    a. Use technical terms appropriate to the level of your audience and be observant of audience reaction to determine when you should elucidate further and define those terms.

5. Make your text large enough so that it can be easily read from the back row, use different fonts and enhancements for headings and text, and always use medium to bold face print so that your text doesn't disappear in the light of the overhead projector.

6. The rule of thumb is no more than six (6) bullets per overhead. This is because the audience can get distracted when faced with a large number of items to reference. An easy method to keep the audience focussed on the point at hand is to use a sheet of paper to reveal only the text you wish to have showing.

7. Include charts, graphics, or illustrations where appropriate to add additional emphasis.

8. Black on transparent is fine but can get boring to the audience. Use colors when possible to increase emphasis and draw the audiences attention; preferably no more than 3 colors for text on a single overhead.

a. Using full color background of blue with yellow and white text has been found to increase audience attention.

b. Using one style of background for your presentation will give it a polished look, but it is also acceptable to change styles between topics which not only gives the audience something different to look at, but reinforces their attention on the change in topic.

9. Try to place critical information at or near the top of your overhead to increase audience retention.

10. As the presenter, do not continually turn to look at the screen as if to figure out which one you have up, or where you are in your presentation, or what you should say next. This makes an audience feel that you are unprepared or not concentrating on the task at hand.

11. Put transparencies in frames to block out the excess light from the overhead projector.

a. If you need to, you can write your crib notes on the frames.

b. Always put a mark on the frame to orient it right-side up and centered so that all you need to do is pick it up with your thumb on the mark and place it on the overhead projector.

(1) one of the most annoying and distracting actions for your audience is when the speaker places the overhead upside down, crooked, too far up or down so that text is unseen, or the constant turning away from the audience to check out how the overhead looks.

12. NEVER READ OVERHEADS TO YOUR AUDIENCE. Overheads are meant to enhance the verbal portion of your speech; to emphasize the main points and to keep attention focused on the topic at hand. Reading an overhead word for word makes an audience feel insulted and breaks any connection formed with the speaker.

13. Do not use your hand to point things out on an overhead. Use a pointer or a pen, or use a piece of paper to cover up text underneath the item you are addressing.

14. Leave your overheads on the projector long enough for the audience to read them; if the point you are making takes only a moment, an overhead probably isn't necessary. When changing overheads do it quickly or if you will not be putting another overhead up for a few minutes, try using a "blackout" overhead instead of turning your projector off and on.

15. If you must include a detailed chart as an overhead, try using a blank transparency over it and track the route of your comments with a colored pen so that the audience doesn't get lost. Also, make sure the audience has a hard copy of any detailed chart so they can follow the path of your presentation.

16. There are many good, fairly inexpensive graphic software products on the market. Take the time to try out demonstrations and choose the one that's right for <u>you</u> and your hardware.

17. As you become more experienced in creating overheads, allow yourself the time to be more creative and check out the technology available which can increase the sophistication of the look of your overheads.

HANDOUTS:

1. The jury is still out on whether audience retention is best gained by distributing handouts at the beginning, during, or at the end of a presentation. This must be a judgement call by yourself, based on the type of presentation you are giving and experience.

    a. In my own experience I have observed that when the presentation is basically a listening experience or one of active audience participation, it is best to inform the audience that presentation handouts will be available to them at the end of the presentation. This allows them the luxury of giving your words their full concentration; and conversely,

b. When you are imparting information to which the audience is expected to take notes and expected to retain and use what they are being told, it is less distracting for them to have the handout materials from the beginning. In this circumstance, the speaker must be more dynamic to maintain the audience's attention to the immediate topic and keep them from ignoring your words and reading through the handout.

2. When handouts are to be used in conjunction with note-taking, design the handout so that it leaves space for making notes.

3. Handouts are not Overheads, Overheads are not Handouts. Remember that overheads are meant to enhance the attention and retention and focus of the audience to what is being spoken and should not be designed with the intention of using them as handouts. The information included in each may be the same and should both follow the path of the presentation, but the format of a handout is that of a document, reference manual, notebook, or worksheet.

HOW TO GET THE BEST USE OF YOUR PC's SOFTWARE AND HARDWARE

A **LASERJET PRINTER** which supports scalable fonts and graphics is a purchase which will pay for itself by reducing the time and cost involved in creating documents and graphics.

SCALABLE FONTS are an inexpensive necessity and are available for most laserjet printers. Scalable fonts make it possible to quickly create attractive title pages for documents and for setting off text in handouts. They can also be used for creating overhead transparencies.

TRANSPARENCIES can be printed directly on your laserjet, but ONLY if using transparency paper developed SPECIFICALLY FOR LASERJET use. Other transparency paper (e.g. for plotters, for copiers) is not designed to withstand the intense heat generated by laserjet printers and will damage your printer. Transparency paper specifically for use

in a laserjet can be purchased from most office supply stores. However, since copier transparency paper is much less expensive you may want to print your overhead text onto plain paper, add any artwork desired, and copy it onto copier transparencies with clear or colored backgrounds.

COLOR PRINTING is available on some models of laserjets.

WORDPROCESSING SOFTWARE can be used not only for writing your outline and speech, but also for doing reference notes if you feel that you will need them for referral when doing your presentation. In addition, you can create overheads using your wordprocessing software enhancements (bold face, italic, line draw, etc.) in conjunction with scalable fonts.

GRAPHICS SOFTWARE is designed for creating overheads and 35mm slides in addition to its other uses. The best products on the market have the following features in common:

- Tutorials which are interactive, have beginner and advanced sections, which take under an hour to go through, and which teach the beginner enough to immediately begin using the software.

- Windows and Mouse environment.

- Variety of backgrounds (or templates) available to user as well as capability to create your own background and change it on command.

- Variety of font styles, sizes and appearances to choose from.

- Wide choice of colors to choose from and pre-designed palettes as well as user designed palettes.

- Has a built in library of artwork to choose from.

- Supports variety of plotters and printers.

- Has built in function to format graphics for overheads or for 35mm slides.

- Has pre-set title and body areas which greatly speed up creation of graphics, but allows user to by-pass these settings if so desired.

- Has wordprocessing functions of word wrap, spelling check, and editing.

- Allows user to reorganize pages in different orders from those entered and to insert new pages.

- Has choice of page formats for title page, one and two column bullets, text combined with drawings from library of artwork, blank format (no bullets), and no-format.

- Has function to do screenshows or viewing (graphics appear on full screen in same format as when printed) at user set intervals or on user's command.

- 'Print' command allows user to choose page to print or to print all pages.

- Has built in icons to use instead of going through command menus for advanced users and allows for user defined icons.

- Supports import of information from wide variety of software, including spreadsheets, databases, and wordprocessing.

PEN PLOTTERS - those with at least 8 pens offer greater creative use as the user can work with more colors and with different thicknesses of the same color. NOTE: It is important that the user decide on a basic palette from the beginning and base designs on that palette. Otherwise the user(s) must retain information on the specific palette used for each document created. When using lighter colors, it is useful to be able to outline letters in black or another dark color to make it show up clearly.

PAINTJETS - the technology advances every year as to resolution, colors, and speed. If doing full color backgrounds, you will need to have extra megabytes of memory; the newest paintjets already have additional memory. Be careful to purchase ink and paper supplies specifically manufactured for your model and model number as the market initially referred only to a Paintjet and to Paintjet XL's. Office suppliers have only recently become aware that there is more than one generation of the XL on the market and each requires a different type of ink cartridge.

ELECTRONIC PRESENTATION OVERLAY PANELS - are LCD panels which are attached to your computer by a cable and placed onto a high illumination (minimum 3000 lumens) overhead projector. All LCD panels will display what is on your computer monitor. Some can also display a video with the same resolution of a wide-screen TV. There are both monochrome and full color overlay panels available.

The monochrome (one dark and one light color) is fine for text documents and for spreadsheets, and for some types of software demonstrations and training. The full color overlay panels are perfect for showing overhead graphics without having to use transparencies. Both types can be used with full size PC's and with laptops, but you must purchase the correct cable with an adaptor that works with your computer (e.g. EGA, VGA). If it is not built-in, you may need to purchase a 'splitter' to connect the disk drive to both your monitor and the panel. The 'splitter' allows simultaneous viewing on the monitor and the overhead projection.

The full color overlay panels come in a variety of models ranging from $2,000 up to $10,000. The feature which affects the price is whether the panel is a passive-matrix (low end) or an active-matrix (high end of technology). The market is changing rapidly, and the same panel can differ in price by hundreds of dollars, so a judicious period of shopping around is recommended.

Even though the initial cost outlay is high, these panels can pay for themselves in just a few years by discounting the cost of plotter and printer ink and transparency supplies, and the cost of the work-hours spent on printing the overheads. The best reasons for purchasing and using the overlay panel is the ability to make last minute and on-site changes and additions to a presentation package and the ability to quickly and easily reformat or edit a presentation to the specific needs or characteristics of the audience.

A note of caution - if you do purchase an LCD panel, remember that LCD stands for Liquid Crystal Display, and all liquids have a freezing point. So NEVER leave your LCD panel unprotected in freezing weather, or it will no longer work, ever again.

EXTERNAL RESOURCES FOR CREATING GRAPHICS and HARDWARE PURCHASE/RENTAL

If you do not have the time, the inclination, the software, or the hardware to create your own presentation materials there are many commercial businesses which will provide this service and/or equipment for you. It will be your call to determine whether this recourse is cost effective for you over the long run rather than doing it yourself, hiring or assigning someone to do it for you, or purchasing the software and hardware which will meet your requirements.

Check your phone company's 'yellow pages' under GRAPHICS, under AUDIO-VISUAL CONSULTANTS, and under AUDIO-VISUAL PRODUCTION SERVICES for businesses which handle presentation graphics. These businesses use a wide variety of graphic software and plotters, printers, and 35mm slide processors, and can do the entire job from scratch or just the printing side of it.

For businesses which sell, rent or lease audio visual equipment such as electronic presentation overlay panels, overhead projectors, etc., look under AUDIO-VISUAL EQUIPMENT - DEALERS and AUDIO-VISUAL EQUIPMENT - RENTING AND LEASING.

And, of course, for computer software/hardware, plotters and printers, call your HP Sales Representative, your RUG vendors, or any computer software/hardware retailer.

FINAL WORDS OF ENCOURAGEMENT

If you have never given a presentation for your Regional User Group, give the idea of giving one some thought, and do it. If something is of interest to you, it is sure to be of interest to others. Don't think of your presentation as a lecture or a do or die situation; we are members of this user group to learn about what other people are doing and how we can use their experiences to make ourselves better at our own jobs. So instead of just sharing with a couple of people over dinner, try sharing with the whole group. We'd all love to listen. And, if you feel an hour is too long, get with some fellow RUG members and put together two or three short subjects of fifteen to twenty minutes and present it as a package deal.

If you haven't spoken in front of a group for a while, try easing into it by speaking up at staff meetings, introducing yourself to people at a social gathering, and making small talk with the person in line at the supermarket. Take the opportunity to state your thoughts at your next local council meeting or community organization meeting. The more you speak up in a variety of settings the more comfortable you will feel giving a prepared presentation.

I highly recommend attending a professional training course that requires the participants to prepare and deliver a five to twenty minute presentation with feedback from both the instructor and the audience. This type of course offers a supportive environment for the beginner and will reinforce the best presentation habits of the experienced speaker.

As you become more experienced in giving presentations you will evolve your own method of preparation that fits your personal working habits, but for the actual presentation never forget the Golden Rule of Presentations:

-- SPEAK UNTO OTHERS --

AS YOU WOULD HAVE THEM

-- SPEAK UNTO YOU --

Paper # 7015
Exploring Client / Server Computing

Robert M. Davia
Hewlett-Packard Company
W-120 Century Road
Paramus, NJ 07653
(201) 599 - 5171

While client / server computing offers an alternative systems strategy that could be advantageous, writing and talking about it is a challenge because of the numerous definitions of the concept. Everyone who implements it, every company that offers a product, and every industry analyst seems to redefine it, often in ways that suit a specific interest. This paper will review the various things that client / server has been used to describe, explore the motivations behind the implementations of technologies that support this kind of computing, present some guidelines for where to implement it, shoot down some common myths that are invoked to derail client / server implementations, and close with some advice on how to succeed with your first client / server application.

The term client / server has been used in what I see as five generic ways, which are shown in Table 1 on the following page. Each of the five kinds of client / server are characterized based on where the three parts of the application take place -- the user interface, the execution of the program logic, and the access of data. To put these terms in context, the traditional application (or system) that uses terminals to access a host could be described as having the terminal as a client and the host as the server. In this case, the terminal provides minimal functionality -- only displaying the character strings (and maybe graphics in some cases) that the host sends. The host does all the work: the generation and control of the user interface, the CPU processing for the application, and all I/O for disc and printing.

Client / Server Technologies

I/O Redirection. The simplest and most widespread software that is called client / server is a Network Operating System (NOS), like Novell, LAN Manager or Banyan. In their original implementation the NOS provided an I/O re-director, which allowed disk and line printer I/O to be invoked on the user's PC (the client) but actually executed on the LAN server. This level of client / server is not very granular; it takes a complete operating system service and transports it to another location. Other mechanisms to be discussed later provide for finer granularity, allowing a portion (or an instance) or a service to be moved. [In current versions of a NOS, not only can I/O be redirected, but execution of an application can also

Configuration	Client Functions	Server Functions
I/O Redirection	Full User Interface Application Processing Local Disk & LP I/O	Remote Disk & LP I/O
Remote Windows	Display User Interface	Drive User Interface Application Processing Data Access
Data Base Server	User Interface Data Editting Application Processing	Data Base Query
Application Server	User Interface Data Editting	Application Processing Data Base Access
Compute Server	User Interface Some Data Access Some Appl Processing	Some Data Access Some Appl Processing

Table 1. Summary of Client / Server Implementation Technologies

be exported to the server, which makes the server much more like the traditional host although it is LAN attached rather than RS-232 attached.]

Remote Windows. Another kind of client / server is remote windows. In this case the client actually executes the user interface, while the server must drive the user interface, execute the application, provide access to the data base and other system resources. Examples of this kind of technology are X-Windows, Motif, OpenLook, and NeWs. These typically require a robust connection between client and server because the traffic to support a sophisticated user interface is generally high. The advantage is that with a relatively inexpensive desktop, the end-user has access to a wide variety of servers with strong graphics capabilities.

Data Base Server. Data bases can also be obtained in client / server mode. Applications developed with this technology can have the client processor execute the program logic, data edits, and control the user interface, while the server simply does the data base and disk I/O. All of the commercial data bases (Ashton Tate, Oracle, Ingres, Informix, HPIMAGE, etc.) today offer such an interface, originally based on the TCP/IP protocol stack, but some now also supporting IPX and Net Beauii. Further, in addition to tools provided by the data

base vendors, there are many such tools from third party suppliers like PowerSoft, Gupta, etc., and this selection of tools make this form of client / server a very popular one.

Application Server. The application server is similar to a data base server, but more generic, and is built on a programmatic interface like BSD Sockets, Named Pipes, SPX, or APPN. Whereas the data base server has an interface that supports making only data base calls, these APIs provide a generic message passing mechanism. A complete transaction could be passed from one processor to another; or a command from a parsing program to a task-oriented program; or a block of data from a pre-processor to a error-handling routine. The increased flexibility in the communications link makes this technology more of a peer to peer implementation; there need not be a clear hierarchy that places one processor in the position of only responding to requests or of only generating requests.

Compute Server. Still another version of this technology is that of a compute server using a Remote Procedure Call (RPC). Instead of simply passing a message, or a block of data, RPCs allow for being able to divide a task into subprograms (called procedures) and invoke these procedures on the appropriate computer for execution. The Open Software Foundation's Distributed Computing Environment (DCE), Sun's Open Network Computing (ONC) or HP's Network Computing System (NCS) are all examples of this. In this environment a program could be logically divided into pieces, so that the pieces execute on separate processors cooperatively. Pieces could exist on multiple machines for redundancy's sake, or they could be placed to take advantage of either hardware capabilities or data co-location. Without a client / server implementation, a program must either execute on the same box that has the data to be manipulated, or a way must be found to copy the data to the right processor and then copy the updated data back; with an RPC implementation, procedures can be located on the processor that houses the relevant data. This allows a program that needs access to multiple different data bases to run in pieces, each piece executing on the processor that can most efficiently provide the data for that code.

Each of these technologies is a type of client / server computing. Simple I/O redirection is not as granular as the APIs (data base server, application server, and compute server). The APIs allow for fairly precise division of functionality across processes (and across computers as well), although they require more programming effort to implement.

What client / server implementations do is break an application into multiple pieces which may run on one or more computers. The common pieces in all of these views are:

- the network -- a high speed mechanism to link cooperating processors;
- a desktop -- an individual's resource (the client);
- a shared resource -- a community resource, be it data base or CPU (the server);

- some software that links these resources together.

I'd like to emphasize the point that client / server does not require multiple machines. In my mind, client / server is a way of making code modular using tools that would allow the various pieces to run on separate hardware; but it could run equally well on a single platform. The most common implementation is across a network covering multiple heterogeneous platforms, and without that goal in mind the whole process of breaking the code up may be without merit. Still, client / server is not defined by the configuration of hardware near as much as it is defined by the software architecture.

Given these pieces, the naming convention becomes obvious. A file server is a processor that other processors use to access files. A print server is a processor that provides printing services for other processors. A display server is a node that provides display services for other processors on the network -- and so the terminology appears backwards for an X-Terminal, in that the display server is the node on the individual's desk, and the display client is typically the larger processor in the data center that does all the computing, and shares the display on the end-user's desk.

Motivations for Moving to Client / Server

Now that we have a handle on what client / server can be, we shall next look at why the industry is excited about it. The most frequently given reason is based on cost. Because PCs and workstations offer inexpensive computation compared to CPU cycles in mainframe-class hardware, MIS managers have been able to convert from host-based to client / server and still save money, even after investing in new client hardware. Of course, the higher the cost of the host-based solution, the easier it is to cost-justify the client / server solution; and so most of the press coverage for these conversions focus on the large dollar savings that are possible in moving applications off of a mainframe. For this audience, which has already shown prudence by investing in minicomputers rather than mainframes, the cost savings will not be as dramatic, and the justification will have to be made in other areas.

Another reason frequently cited in the press is competition. As companies use IT strategy to improve service, or reduce costs, or speed-up decision making, other companies find that they have to match or beat those productivity improvements. Client / server development promises to reduce the application backlog because the development tools that are available make it relatively easy

- to re-use application logic and user-interface code to speed development, and
- to change the logic or appearance of the application when modifications are required.

And as CASE tools begin to address the client / server environment, not only could the modifications become faster, but the orchestrated delivery of updated software to the various clients and servers could be significantly improved.

Rising user expectations are also pulling these solutions into play. As users get used to the "point and click" world of desktop computing, they become less satisfied with character mode host applications. Users want the flexibility, ease of use, and good graphics not just on their spreadsheets and e-mail, but also on their applications that are currently executing on the host. Add to this the simplicity now of building a LAN -- which for a workgroup you can create in a couple hours and $1,000 with the "LAN in a box" products available in dealers -- and you have departments ready and waiting for client / server.

Of course, it is not just the end users who see advantages in client / server; the IT staff sees that it can be easier to provide smooth performance by distributing applications from one processor to multiple desktop processors. Further, it is easier to provide access to multiple hosts in a client / server environment than it is in a dumb terminal environment, so data centers with multi-vendor systems can reduce connectivity head-aches with desktop systems. And endorsing client / server (and so taking control of the systems that end up at the desktop) provides IT a way to minimize the diversity of desktops that will naturally occur if departments are left on their own.

Getting Started

So, if you want to start with client / server, where do you begin? Do you need to find an application that would really benefit from a graphical user interface? (No, surveys show that only 31% of existing client / server applications use a GUI, although 70% of planned applications include one.) Do you need to discard your investment in existing code? (No, only 30% are re-engineered applications, while 70% are new applications.) Do you need to bet you company on this new technology? (Maybe. 54% of client / server applications are rated as mission critical -- I would think that this has to do with the managerial commitment required to shift corporate gears as MIS changes direction.) [Statistics are taken from Datamation, May 15, 1991, "Client / Server's Hot In New Applications."]

Historically, client / server evolved in the scientific and engineering worlds, not in the commercial environment. To provide the high local processing needed by modeling applications and by applications like CAD/CAM, protocols were developed that allowed powerful systems to work cooperatively. Although with the advent of desktop price wars, putting computational power on the desktop is no longer reserved just for specialized applications.

Next the large concern was to share an expensive resource, like a high quality laser printer, or a drafting-quality plotter. And while this may be true in the short term, the trends seem to be that resources that are in demand, soon fall in price.

Now the key for client / server systems is a community with shared data needs. Whether that shared data is a real data base (of customers, parts, or what have you), or an application that only a department exercises, communities of interest define the structure and implementation of client / server. In most businesses, there are various communities of interest that any one person would maintain. And the client / server implementation can reflect those overlapping communities, for example, with multiple servers each supporting a different audience for a given application.

One other criteria to consider is the ratio of processing to data access. Applications with a high processing to data access ratio tend to apply themselves to client / server environments nicely because they can perform most of their processing on the client, leaving the server to optimize the data access portion of the application. In the extreme case of this, in an application that was purely data access, with no processing, this would be the same as the classic dumb terminal to host environment, which would not take advantage of the capabilities on the desktop. In the other extreme, with high application logic demands, by offloading the processing to the desktop, more users could reasonably be supported by a smaller data base server, and the application would be running on the least expensive processor possible.

On the other hand, there are lots of reasons cited for not moving to client / server technologies. Here are reasons that I have heard, along with my reaction to them.

- *I've got millions invested in my existing solutions.* A migration to client / server is just that, a migration. Existing applications don't get thrown away; the technology for client / server can easily access existing host-based programs. In the best case, the existing host over time is simply re-tooled, as new applications use it not as the only resource, but as one of a number of shared resources. The host becomes a server. Eventually the need for the large host which concentrates so many resources into one box may diminish, which is good given the relatively high price for host resources.
- *I've been successful so far with terminal to host methods.* Yes, you have, but so has your competition. And if they are moving, you should at least consider moving. Further, it may be that the needs of your user community are changing. The services you provided successfully in the past may no longer address the needs that your users present.
- *I've got too many different clients to integrate.* Don't try to start by integrating them all. Especially for a first project, limit the scope of the implementation. Get comfortable with the tools in a simpler environment. And time is on your

side; each month new tools are announced that provide services to more and more different clients.

- *We can't get users to back up data now, and you want more servers?* Yes. All data needs to be owned by some one; some one who is responsible for knowing that the data has integrity, and will take action to fix it when integrity is lacking. If you have data without an owner, you have a problem waiting to happen. By providing servers, and encouraging your user community to take advantage of the disc space that you will manage, you will collect user data in a limited number of places -- so that you need only back up the servers, and not every user's desktop device.
- *I'm going to wait until my software vendor supports it.* OK. For applications that run your business, you may want to stick with a vendor that is delivering a satisfactory solution. But you may want to begin pushing that vendor to offer a client / server version. Or if you some day find a business need that requires client / server, you may want to explore alternative vendors that do.
- *Client / server seems so complicated.* It can be, but isn't everything that we do complicated? Didn't terminal to host seem complicated once? There is a comfort factor with what we already know, and any new subject can seem intimidating. Client / server applications have the same basic pieces that a terminal to host application has: 1) a user interface; 2) application logic; and 3) data access routines. Give yourself some time to learn some new tools, and you will be speaking the new language in no time at all.
- *My network is already too crowded.* The beauty here is that client / server can reduce the demands on your network. If you have desktops today without good client / server tools, then you have some very wasteful activities going on. Users are moving files around when all they want are a few records. And client / server allows you to rationalize your network; by better defining who needs access to what resource it is possible to place resources closer to the user, and so minimize the demands for bandwidth.
- *How do we keep our data in sync when it is on several computers?* If you have got desktops, you've already got data all over and its probably already out of sync. Servers allow you to consolidate and gain some control over the data that users are basing their business decisions on. Now clearly there are issues relating to how to keep multiple data bases consistent, but you have this same problem on a host with multiple data bases, and the solutions will be very much the same.

Now for those who want to pursue client / server, here's some advice on how to get started.

Pilot first; roll out later. The role of a pilot is critical for learning what users want and how to give it to them. Sometimes I think that we in IT often over-estimate the capability of our users to envision the right technological solution to their concerns. A pilot offers both IT and the users an opportunity to apply a new technique to obtaining information to make decisions. It affords all

parties a chance to see what is possible before making decisions about how to implement it.

A pilot is also an opportunity to experiment. And while you want to put a good deal of thought into what you pilot, you also want to be open to changes during the pilot. Once a working system is in place, users will suddenly see new ways to utilize it. You may see new ways to deploy it. Allow yourself the freedom to 'fail'; that is, what you pilot does not have to be what you deploy. In fact, if the pilot was a real learning experience, the two *should not* be the same.

Model and size all elements. Right at the heart of application design is the need to define what data is being used, and how it is used, and this is no less important in client / server design. Dividing the transactions into what part takes place at the user's desktop and what part takes place at the server needs to be based on a solid understanding of what data it takes to perform each operation. This will impact what data is distributed, if any, and how much information needs to pass over the network, and how often, which then becomes the basis for the network design.

Define an architecture. The act of defining an architecture forces you to address each area where you have a design choice to make. It also gives you a document that you can use to communicate your needs to your vendors. And it insures that there are no uncovered issues. The current debate is whether a two or a three tier architecture is best. This refers to the three application areas mentioned above: the user interface, the application logic, and the data access. A three tier architecture addresses each area independently, and provides the most flexibility in selecting tools and placing functionality where it makes most sense. (Unfortunately, many tools are based on a two tiered model, bundling the user interface with the application logic.)

Another common version of the two versus three tier debate is centered on the hardware platform that is used. In a three tier model, there are collections of data (in a data base server or in a legacy application), which compromise the top tier. A middle tier provides a flexible interface for combining pieces of data from any of the top tier machines an perhaps supplementing that with additional data or analysis, and passes it along to the desktop tier which provides the user interface. A two-tiered version of this simply removes the middle layer of processors and pushes that functionality into either the desktop or the data base server. The advantage of the middle tier is that it often provides a less expensive development environment (especially if the top tier is a mainframe), and the separation of code and data that it provides can simplify maintenance.

Stick to standards. As in all things, it is usually quicker and easier to take short cuts. Just be aware when you deviate from standards, because the pain

of sticking to standards pays off in ease of growth and expansion. Its a trade off that is made every day, and you need to make it consciously.

Be crystal clear about ownership. All data needs an owner, and so be clear about who is responsible for the applications and the data. Yes, it can be tedious to make operational plans to get the appropriate checks in place to keep the data in a data base accurate. But that data is a corporate resource, and as data processing professionals it is our job to groom the data that drive our businesses. Besides, its easier to keep data clean than try to clean it up after it gets messed up.

Consider your Organization. Today most DP shops are organized by application, and the team owns the entire solution. In a client / server shop it may make sense to organize by specialty, with a user interface group, a data base group, and an application logic group. From these groups, you may still assemble a team that will build an application, but the application will be built on top of an infrastructure that is shared by many other applications, and there ought to be savings in the re-use of parts that are shared with other applications.

Remember the auditors. Security issues do not change much -- you still will have to deal with issues around who can access what data, and when. There will be passwords, and keys, and dial-back modems. These are facts of life today, and simply need to be incorporated into the procedures and software that you implement.

Budget for training. As the company becomes familiar with this new technology, everyone will need some training: your staff, in both the development and operations areas; users who need to maneuver through a new environment; and managers who must come to terms with what this new tools means to the way they do business.

Client / server may not be the "be all and end all" of computing, but there are several compelling reasons for its development. Done appropriately, it can help businesses manipulate the information they need in a timely fashion. It offers a way to utilize the tremendous computing power available at very attractive prices in a way that still allows for a central staff to tend to data integrity issues. It should be viewed as yet another tool in your toolbox, to be used as appropriate. "When you only have a hammer, everything looks like a nail." Hopefully, equipped with some client / server experience, not every application requirement will look like a centralized data base.

Paper Number: 7016

HP OpenODB in the Oil and Gas Industry

Douglas Dedo

Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino, CA 95014
408-447-7726

ABSTRACT

The oil and gas industry created an organization called the Petrotechnical Open Software Corporation (POSC) that is defining the software platform on which their next generation of applications will be implemented. POSC's main objective is to lower information technology costs by making it easier to independently develop software that works together. An average of 60% of an exploration employee's time is spent looking for data and up to 70% of computing time is required converting it into a usable format. New solutions need to address quickly accessing all data, making it consistent, and sharing it between applications and organizations. This task is challenging due to the vast amounts of complex data stored in many different places and due to the need to support many custom, legacy systems. These challenges are similar to those faced in the telecommunications and other industries.

POSC has selected HP's object-oriented database, OpenODB, as one core technology component that will be part of their initial software platform implementation. This paper will describe the complex challenges the petroleum industry faced that led up to the creation of POSC, the objectives they are addressing, and how OpenODB helps to support their needs.

The vision

Let's take a look at an ideal environment for getting oil exploration information and sharing it with business partners.

Philip is reviewing data gathered from oil wells that were recently drilled. Philip's company is evaluating the potential of this new site. He and Allan, a geologist from another company, are sharing their respective data to get a clearer picture of the rock formations. This exploration process is risky. The cost of pursuing a dry well, one that produces no oil, can total up to one billion dollars. The rewards can be substantial.

Philip uses a number of powerful applications on his workstation to analyze the data. He looks at the fossil history in the soil samples, estimates the period in time for each layer of sediment that the drill passed through, reviews the seismic readings taken in the area, and compares his findings with similar data from other test wells around the world. After reaching some conclusions about the attractive potential of this area, he contacts his colleague and compares notes. Allan views the key information from some new tools and shares Philip's excitement.

Philip and Allan pass their exploration data onto the employees in the Production group who will do additional analysis to verify the geologist's conclusions and work to optimize the process for removing the oil from the ground.

The reality

Although this imaginary scenario sounds simple, in practice, exploring for and producing oil is a time consuming, awkward, and costly process. Getting the data takes time. On average 60% of an exploration employee's time is spent looking for data rather than doing useful work. Getting the data to be consistent is awkward. Because the data was captured and computerized in many ways the information is stored in dissimilar formats and distributed across different computers. In a typical company there are as many as 50 to 100 different ways of storing the same information. Up to 70% of the computing time is spent translating the data to a usable format for one application. Sharing the data is costly. For every dollar spent developing a new oil and gas application, $1.50 to $2.00 more are required to integrate it into

today's environment so that the information can be shared between complementary tools (see Figure 1).

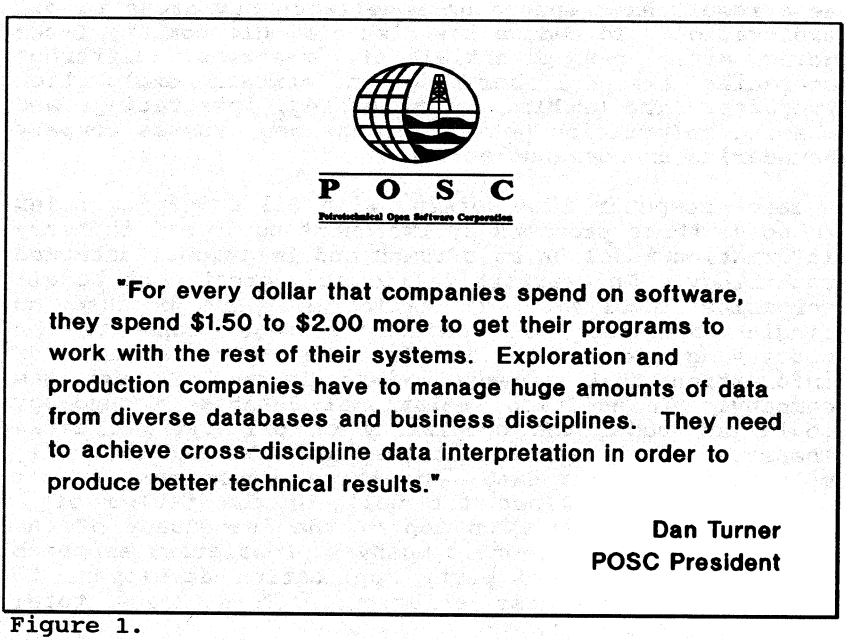


Figure 1.

The problems outlined here only represent the process within one department. Today, after an Exploration group has gathered information by drilling a series of expensive test wells and has determined that a site should be handed over to the Production group to move into the next phase, a crime takes place. The exploration data is thrown away and new production data is gathered by drilling a new set of test wells. Since the view of the data that satisfies the needs of an exploration geologist are different than that of a production engineer, it is often easier to start from scratch than try to translate the existing data, even though much of the core information is the same.

Oil and gas companies have been stoic to put up with this costly procedure for such a long time. But recent world events have made this painful process too visible to ignore. The petroleum industry has had severe competitive pressures placed on them. Overproduction has driven prices very low. Environmental protections have made it too costly to explore or have stopped production altogether in sensitive areas. Most of the easily accessible oil fields have been found and some

are going to run out in the not too distant future. The former Soviet Union has been courting sources of hard currency to fill some pressing financial needs and as a result have opened up some large new areas to oil exploration. To reduce the risk a single company faces going after new potential oil reserves, different companies are collaborating on certain exploration projects. The problems of accessing, integrating, and sharing information between groups now crosses company boundaries and demands action.

A large computer firm working with oil companies tried to solve these problems by implementing an oil industry information model on mainframes and relational database technology. This effort failed at a great cost to all companies involved. It failed in part because no single computer has enough storage capacity or processing power to handle the vast amount of information that already exists in an oil and gas company. In addition, relational database technology could not model the complexity of oil and gas data. The extensive arrays of data captured for one oil well, multimedia seismic data, and three dimensional models of oil fields do not fit well in the tables of a relational database. On top of the inadequacy of the chosen technologies, one company's proprietary approach did not entice third party application developers to complete the business solution. This keeps total solution costs high.

The Exploration & Production (E&P) industry also tried to respond to the escalating costs of maintaining and supporting E&P computing technology by beginning to buy applications from software vendors instead of building their own. This approach helped to spread the cost of software development and maintenance across a larger user base and thus was a practical way of continuing to fund the development of increasingly complex application programs. But data management problems and associated support costs grew worse. Even those companies that had established corporate data files to manage their technical data had significant problems since each purchased application carried with it a unique internal format for storing information. The corporations themselves had to bear expensive hidden costs of developing software to translate from the applications' formats to the corporation's own format. Even if they abandoned the goal of effectively managing their data and let it be scattered into the internal files kept by the various applications, they still had to develop software to translate from one application's format to another's. No application was an island working independently of the others. The burden of creating ways for data to flow between applications

still fell on the E&P companies. At the same time, the potential market for application vendors was constricted to just the largest customers who could afford the integration costs. What was a burden to the E&P companies was a barrier to vendors wishing to sell computing technology.

Another hidden support cost associated with buying application programs was the amount of user training required before E&P employees were proficient in using the applications. Since programs were developed independently of each other, the basic rules for operating the programs were different. With the increasing diversity resulting from buying technology from multiple vendors came increasing confusion and inefficiency. Companies that had tried to reduce their expenditures on software development and maintenance by purchasing applications instead of developing them in-house still had a major business problem on their hands - exorbitant computing technology support costs. The buy-not-build approach by itself did not significantly reduce information technology costs.

Leading oil and gas companies take control

As a result, five visionary oil companies (British Petroleum Exploration, Chevron, Elf Aquitaine, Mobil, and Texaco) founded a non-profit organization called the Petrotechnical Open Software Corporation (POSC) with a goal to provide a common set of specifications to implementors of E&P technical computing systems that, if followed, will allow data to flow smoothly between products from different organizations and will allow users to move smoothly from one application to another. POSC's members include more than 70 leading petroleum companies, service companies, hardware and software vendors, along with government, academic, and research organizations from ten different countries. POSC members believe the scenario at the beginning of this paper can result from their efforts. Since the technical data encompassed by POSC's specifications will span the entire spectrum of E&P technical activities, colleagues from different disciplines will be able to share data with each other. The major business benefit of POSC's effort is to reduce the cost of locating and producing oil. They have little to lose and a lot to gain through this endeavor.

To reach their objective, POSC is defining a set of publicly available specifications for a Software Integration Platform, or SIP. The SIP will cover all aspects needed for E&P companies to more efficiently manage their technical data resources. This includes

defining base computer standards, an E&P data model called Epicentre, the application programming interface (API) to access the data model, a format for exchanging data between different systems, and specifying guidelines for software user interfaces. To avoid the pitfalls of previous attempts to solve these problems, POSC is taking a distributed, open systems, and object-oriented approach to the overall architecture. Since a mainframe-based approach did not suffice, breaking the solution into smaller, manageable pieces that run on different computers is a pragmatic alternative. These distributed pieces are linked together by a common information model and standardized methods for exchanging data. Instead of relying on one vendor to provide a complete solution, POSC is taking an open systems approach that integrates the best-in-class components from different vendors. And object-oriented technologies are being used to model the complex E&P information that did not lend itself to relational database technology.

Besides creating standard technical specifications to allow new software to work together better, POSC is doing a range of activities to get companies to quickly develop applications that adhere to the standards. They are supporting the rapid development of two sample (reference) implementations of their data model and API to jump start the application development process. Interoperability labs with computer hardware and software have been set up in Houston, Texas and London, England to give POSC members hands on experience with the reference implementations. Training classes are being provided on different aspects of the specifications. And conformance tests are being developed to verify that new applications do follow the standards defined by POSC.

What has POSC accomplished so far?

POSC published the first version of its Base Computer Standards in late 1991. It identifies vendor-neutral standards in seven areas. The standards discuss hardware technology that spans from the smallest desktop to the largest supercomputers.

An extended version of the EXPRESS data modeling language was chosen to describe POSC's E&P data model. EXPRESS is an International Standards Organization (ISO) draft standard which has also been used to describe the information models of other large organizations like the Computer Aided Design (CAD) Framework Initiative.

Early drafts, or snapshots, of the POSC data model were published starting in early 1992. The first complete version of Epicentre was published in the first half of 1993. More than 4000 E&P and technical data items have been defined in the data model that include such areas as drilling, geology, geophysics, cartography, production, reservoir, and wells. It is important to note that when the work started on the data model, everyone expected it to be implemented using relational database technology. After six months of effort developing the model, people stood back and realized that they had really created an object-oriented data model. Important structures like three dimensional earth models and huge arrays of data for oil well logs do not all fit into a relational table. Epicentre not only described the data used in exploration and production but also typical functions that would be performed on that data. For example, in addition to defining the depth and location of a well, the unit conversions like meters to feet and Cartesian coordinates to latitude/longitude could be applied based on the end user's perspective.

For those developing applications that use POSC's data model, a User Interface Style Guide has been developed that covers basic "look and feel" issues like menu selections and scroll bars. With the focus on standards, the presentation of windows follows the Motif format.

Working with the Society of Exploration Geophysicists and other organizations, POSC has adopted standard formats and definitions for exchanging seismic and other E&P data between companies and computing platforms. The base selection for an exchange format was the American Petroleum Institute's Recommended Practice 66 (RP66). Continuing in this standards oriented role, POSC has become an active member of other industry standards organizations like the American National Standards Institute (ANSI) for their work on SQL3 (the next version of the structured query language for databases) and the Object Management Group for their work on distributed object-oriented services.

A request for technology was published in January 1992 describing the needs of the Epicentre data model and the API to access the data. Submissions were received from 29 different vendors. After evaluation, the submittals were grouped into four categories of technology (extended relational databases, pure object databases, hybrid object-oriented databases, and E&P application view interfaces submitted from different oil and gas companies). The hybrid object-oriented database approach was considered the best overall fit

for POSC. The two databases that fell into this category were HP OpenODB and UniSQL. POSC initiated the development of a reference implementation of Epicentre on each database. Two early releases of this work have already been completed with the full 1.0 Version to be done by the end of August. POSC members can experiment with these early implementations within the Interoperability Labs at POSC's offices.

What remains to be done?

The "plug-and-play" applications that this industry desires can now begin development using the new implementations of Epicentre. Both petroleum companies and application software vendors that support the oil industry are beginning their Epicentre-based work. While some oil companies will give buying preference to Epicentre-based solutions when there is a choice, some companies hope to only purchase POSC-compliant applications after a few years. This is based on their hope that a complete set of POSC-compliant applications that span the needs of E&P professionals will be available.

POSC plans to develop a series of tests that can be used both by companies within the industry for their own internal testing and by POSC for testing compliance with the specifications. Companies wishing to sell POSC-compliant software will be able to submit their products to POSC for certification. POSC, or one of its licensed agents, will run the appropriate POSC test suite and, if the compliance tests are successfully passed, vendors will obtain permission to advertise that their product is officially POSC-branded.

POSC's efforts are not being embraced by everyone. Software and hardware vendors that have strong positions within the oil and gas industry feel threatened. Inside the petroleum companies themselves there are those resistant to change. However, the systems in place today are too inflexible to respond to all the changes the industry has faced and continues to experience. Some needs have not been addressed at all. And the costs have continued to mount. This is a clear opportunity for new technologies to fill a vacuum much like relational databases did when they followed hierarchical databases.

What is this new technology?

The hybrid object-oriented database technology is an evolutionary step from relational database management

systems (RDBMS). They physically build on top of relational databases to leverage the benefits of these powerful systems. What they add is a more powerful way to model an end user's view of the world.

Originally applications stored their data in the computer's file system. When commercial applications drove the need to share this data, network-model and hierarchical databases emerged. The internal model was data sets linked by a hierarchy or network of relationships. The need for more flexible decision-support led to the industry standard SQL language and relational databases. Internally, these systems define data in a table (rows & columns) model. There are no relationships modeled within the database however. Relationships are captured in application code or are hand-crafted by an end user in an ad hoc way. The need driving the interest in object-oriented databases is a more complex decision-support environment where the relationships between the data, and functions performed on the data, are as important as the data itself. The internal database model is built on user-defined types of information called objects.

An object, simply stated, is business information that is simulated or modeled in a computer. Objects model business information from a user-oriented perspective rather than a machine-oriented view. The benefits of using objects are to make it easier for developers and end users to manage and keep track of diverse business information while dramatically reducing software development and maintenance costs.

About OpenODB

OpenODB is a commercial-grade object-oriented database designed to support hundreds of users and very large amounts of data. This database offers customers data-center security and maintenance, a query language, 24-hour availability and the ability to run the database while simultaneously backing it up.

HP's database approach allows users to progress naturally to object-oriented technology while still accessing and using existing applications. This is because OpenODB's object-oriented structured query language (OSQL) allows users to retrieve information regardless of where it is or how it is stored. OSQL has been used as a basis for defining a POSC application programming interface (API), which facilitates storing and accessing E&P information. Major object-oriented components of OSQL were incorporated early in 1993 into the current ANSI SQL3

draft standard and Hewlett-Packard will continue to work closely with the many other companies involved in support of this formal industry standard setting activity.

Integrating legacy code and data

Creating a complete information model, like POSC's Epicentre, and mapping it into an object database is the first key step. Once the model is designed, getting at the actual data presents a significant problem. Recognizing that most corporate data today is stored on different systems across a wide geographic area, it is necessary to have a flexible way to map from the business model to where the data is physically located. This is necessary since not all existing data can be moved into a new database because not all existing applications that use the data can be (or should be) re-engineered immediately. It is also not good practice to copy the legacy data as duplicate copies create a management nightmare.

OpenODB contains a mechanism called "external functions" that provides developers with a way to exit from the database to get access to any data throughout a company-wide network. This ability to model complex businesses and map that to any type of data anywhere has been one of the key reasons large commercial customers are purchasing OpenODB. One business benefit is that it allows a company to do an evolutionary approach to new object-oriented technology that coexists with existing legacy systems. Figure 2 shows an architectural picture of this type of data integration application. The external functions can also be used to access existing code or tools to leverage from previous software development. For example, if a complex algorithm has been developed in FORTRAN for manipulating seismic data, it can be accessed by OpenODB instead of writing the code again.

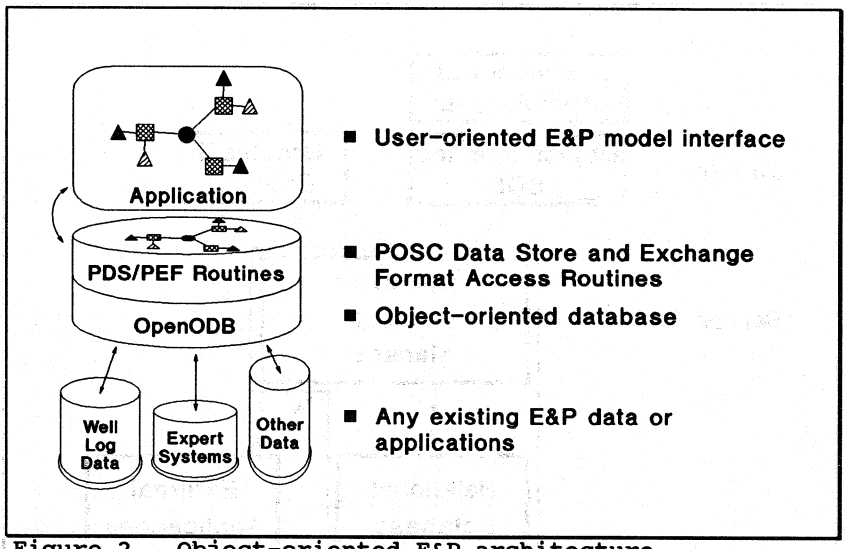


Figure 2. Object-oriented E&P architecture.

Multivendor, open systems perspective

OpenODB was architected as a layer of software, called the Object Manager (where all of the object-oriented aspects of the product are handled), on top of a relational database, HP ALLBASE/SQL (see Figure 3). This was done so that HP could deliver an industrial-strength, commercial object-oriented database to the marketplace as quickly as possible. This was also done so that HP could license this software layer to other companies. In this way, the technology becomes available from a number of vendors on the popular hardware platforms.

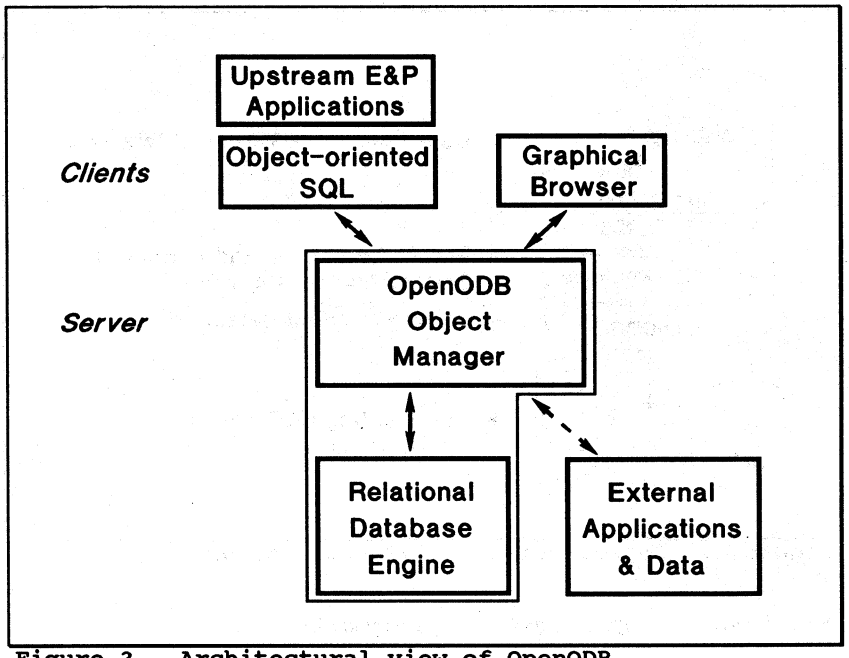


Figure 3. Architectural view of OpenODB.

The first business partner with plans to port OpenODB onto another relational database is Informix. Informix has the largest installed base of relational databases in the UNIX world today and they have the largest number of distributors on a worldwide basis of any independent relational database vendor. They are in the process of porting the OpenODB Object Manager on top of their relational database Informix-OnLine and they plan to port it across different hardware platforms.

Continuing a practice found with HP's SoftBench and OpenView programs that sell and support software on other hardware platforms, the OpenODB client software is being ported to other workstations including IBM-compatible personal computers and Sun workstations and will be available directly from HP.

Extensive tools program

Because of OSQL, programmers can use third-generation languages such as Ada, C, COBOL, FORTRAN, PASCAL, and object-oriented languages like C++ and Smalltalk to write OpenODB applications. It is also possible to use any development tools that automatically generate this

code when creating OpenODB applications. For example, Interface Architect (UIM/X) is a tool that builds Motif-based graphical user interfaces and it generates C code that can call OpenODB.

The OpenODB product includes a tool for developers called a Browser that makes it possible to inspect the internal structure (schema) of the database as well as the end user information stored inside. OpenODB also includes utilities for database administrators to manage the database (e.g. on-line backups) as well as facilities in OSQL to programmatically get information about the database schema. This functionality is particularly useful for tool builders.

HP is working with a number of third party software tool vendors to more tightly integrate their tools with OpenODB. The first business partner was Information Builders, Inc. (IBI). They are connecting their FOCUS report writer, which has over 1 million users, with OpenODB. HP has created a connection between OpenODB and IBI's EDA/SQL (Enterprise Data Access/Structured Query Language) tool that can access more than 50 commonly used databases over many different networking protocols. CGI Informatique, the largest Integrated CASE (I-CASE) vendor in the world is building its next generation of tools and repository to work with OpenODB. Informix has a powerful set of object-oriented tools that they plan to connect with OpenODB. STEP Tools, Inc. is updating their EXPRESS language-based tool to automatically take an information model and generate OpenODB schema. Hitachi's ObjectIQ object-oriented development tool is also connecting to OpenODB to automatically generate OpenODB schema. There are other tool vendors working with OpenODB on projects they feel are such a competitive advantage that they are not willing to talk about them publicly yet.

OpenODB has been encapsulated into SoftBench so that developers using this environment can see the object-oriented functions available in the database.

Finally, HP has connected the Smalltalk programming language with OpenODB so that Smalltalk tools can be used with OpenODB.

Distributed object computing

HP was one of the first companies to realize the importance of objects. There has been extensive experience within HP in commercializing object and distributed computing technology. The research work

that formed the basis for OpenODB started in 1984 with the Iris project.

In July 1992, HP announced its Distributed Object Computing program, which uses OpenODB as its core data management component. This program allows HP to give customers with large, complex organizations several migration paths to distributed object computing, while protecting their existing software and hardware investments.

The other major components within this Program include C++ SoftBench as a software development environment, HP VUE (Motif) as a user interface, HP OpenView as the network and systems management environment, Distributed Computing Environment (DCE) for inter-application network communication, and HP Distributed Smalltalk as a complete implementation of the Common Object Request Broker Architecture (CORBA) technology from the Object Management Group (OMG). Both the definition of DCE and the Distributed Management Environment (DME) standards within the Open Software Foundation (OSF) were based on HP's technology. The CORBA standard from the OMG came from object technology work at Hewlett-Packard.

Where does this technology fit in the oil industry?

The user's interface into the computer is incorporating more graphics making it easier to understand and use the applications. For oil and gas professionals one common interface graphic is a map. On this map would be basins where oil exploration or production was being done. By zooming in on a particular oil field, more detailed information would appear on the map about the oil wells. By zooming in on a particular well, other graphical pictures would appear by applications displaying the layers of rock sediment found when the well was drilled or even pictures of fossils and pollens found in the core samples. Three dimensional seismic models could also be brought up for the immediate area being explored.

The initial application discussed above, using the map interface, is called a Geographic Information System (GIS). When the business information is displayed with a GIS application, there are many relationships one can associate between different objects on the screen. For example, a geologist might want to find all oil wells within five miles of an airport (for easy access) that have a particular fossil in a specific type of soil (to compare recent findings against proven oil wells in other parts of the world). To model this type of complex relationships, and to allow the geologist the

ability to craft an ad hoc query to get access to this information quickly, requires an object-oriented database like OpenODB. To meet these needs, OpenODB can model relationships between GIS information within the database. The OSQL query language can then be used to build a request for information based on any combination of these relationships.

Since people who do oil exploration work spend up to 70% of their time just looking for data, a geologist's workstation that integrates data from different sources can play a valuable role. OpenODB can access any data available on the network using the external function capability and make it look like an OpenODB object. By creating a mapping between the end user's business model and relevant data that exists around the company, information can be brought to the end user much more easily than happens today. Legacy data that is often available but not easy to pull together includes information about oil wells, reservoirs, and oil fields; information about the geography including who has current drilling rights for specific parcels of land; research information done on techniques for extracting oil under different pressure and mineral conditions; and related data at a business partner's company. Getting a global view of all this data can save time and improve the quality of business decisions.

Another application fit for object-oriented databases is where two people doing different jobs need to work with the same core data but with different business views. The same data used by those exploring a new area for oil can be applied by those who manage the next phase of the process that produces the oil which then goes to the refineries. OpenODB makes it easy to define different end user views of information directly in the database. At a minimum, this can cut expenses related to drilling new test wells to get information for production experts that duplicates earlier test well data for exploration professionals. It also can reduce the costs associated with storing huge amounts of redundant data.

There has been an increased awareness worldwide on the need to safeguard the environment. The oil industry makes a large effort to avoid oil spills or other disasters. However problems do surface from time to time. One type of application using an object database is one that can assist in the management of an emergency situation. Complex emergency response processes can be stored in OpenODB to support this work. If a spill were to occur in the North Sea, for example, the object database can be queried for the

closest resources available to the spill site and then have the database activate the process which calls and notifies this facility about the problem and where it is. Resources like helicopters to rescue people, boats to mop up the spill, and agencies that tend to injured animals can all be tied into the object database model.

Knowledge-based systems have been developed to a point where they can play a useful role with object databases. A knowledge base that takes in information about plant fossil fragments and uses it to identify the species or exact plant supports the process a geologist goes through when analyzing oil well core samples. Also applying common business rules while inputting this new fossil information into a computer can potentially eliminate inconsistent data. This saves the cost and time of cleaning up the data later.

OpenODB solving problems in the Oil and Gas industry

The oil and gas industry has been faced with escalating costs associated with computing technology costs. This has become significant with the increased competitive and pricing pressures within the petroleum industry. Neither approach of building custom software in-house or relying on applications developed by outside vendors have reduced the cost of technology. In fact, the problems appear to have become worse.

A group of oil companies have formed POSC to take an industry-wide approach to addressing the problems. By standardizing on a common information model, they hope that applications that are developed using the model will be easier to use and thus require less training for employees, will incorporate a more complete view of all relevant business information, will allow for better collaboration between small, cross-disciplinary teams, and most importantly will cost less due to a broader plug-and-play marketplace.

OpenODB, a hybrid object-oriented database, has been selected by POSC to implement a reference implementation of the POSC data model. OpenODB brings both the powerful information modeling capabilities of object-oriented technology as well as the commercial database capabilities like integrity, security, and high availability found in relational databases. This is because the object database software is built physically on top of existing relational databases. This reuse of robust, production-tested relational database code dramatically reduced the time to bring this object-oriented technology to the marketplace. It also provides the open systems structure making it

possible to deliver the technology on many different computers.

Oil and gas companies interested in finding solutions to the high computing costs of today can find a promising new approach through the work being done by POSC. Gaining experience with OpenODB's implementation of POSC's data model can begin the process to better data access and management at lower costs.

Douglas Dedo is the product marketing manager at Hewlett-Packard responsible for HP's object-oriented database program. Doug has held product marketing, marketing communications, and software development engineering positions within Hewlett-Packard's computer networking group. Previous to HP, he worked on networking and database projects as a software development engineer in the United States and Europe. Doug holds a B.A. degree in computer science from U.C. Berkeley.

Paper # 7017

MEETING THE NETWORKING CHALLENGE: Connecting your PCs, Hosts, and LAN Services

Karl Crabs
WRQ (Walker Richer & Quinn Inc.)
2815 Eastlake Avenue East
Seattle, Washington 98102
(800) 872-2829

The Problem

The migration from the asynch world to the network domain is now in full swing. The percentage of devices connected via network links has risen steadily over the last few years and promises to continue to grow exponentially over the next decade. In a "textbook case" the transition to networked computing would be driven by the obvious benefits of the new medium, i.e., faster data transmission, dramatically increased transmission reliability, and simpler and relatively inexpensive wiring schemes. Yet while these are all critical, the real driving force is something quite different.

The most influential single factor in the dramatic shift towards network connectivity is the rise in popularity of PC LANs—Novell, Banyan, and LAN Manager (LAN Man) network operating systems (NOS). The virtual disk capability at the core of these network operating systems—with its file sharing and storage benefits—appears to be something that corporate end users cannot live without. The movement towards LAN-based e-mail and the efforts the NOS vendors are making in the areas of network management and host-server communication are likely to intensify this trend. All of these factors point to an increasingly important role for the LAN server in the corporate computing environment.

The PC LAN has normally been implemented at the workgroup level, where some basic infrastructure must be put in place. PC LANs require Ethernet, Token-Ring or Arcnet connections at the desktop. Network cabling must be brought in, and network interface cards installed on the PC, and network operating systems and their respective servers installed and configured.

Although PCs have gained LAN capabilities, they usually have maintained serial connections to host computers, too. The typical shop has continued to rely on a variety of these connections—direct connections to a host port, or through a dedicated terminal controller, data switch, or, in some cases, a MUX. But even with all of these connections, the user still has difficulty accessing more than one host and even greater difficulty accessing more than one type of host. (For instance, many users at an HP site may need access to an HP 3000 and an HP 9000.) In addition, just maintaining and

supporting these connections is a job unto itself.

The result of these two separate activities is connectivity redundancy. A PC that is on a LAN and has serial connections to a host has both a serial and a network drop. The network manager still must handle two separate configurations. So the question becomes: What can the network manager do to handle communication with these disparate services and their respective connectivity schemes?

The Options

The ideal situation is to provide one wire from the desktop to give users access to the variety of hosts and PC LAN servers on their local and non-local networks. In order to achieve this ideal, all of the hosts and servers must have either a common language or a translating device. Getting beyond the "language barrier" can be achieved in a variety of ways. The MIS environment can be designed so that all the host and servers use a standard protocol suite such as TCP/IP (Transmission Control Protocol/Internet Protocol). Another option is to build gateways or protocol converters between the various devices to allow for "language" translation. A third option is to give the desktop device the multilingual capability to speak the languages it needs when addressing each of the respective nodes. Each of these approaches has its relative strengths and weaknesses, which I will discuss in this paper. But first I should give an overview of the primary languages we are likely to see in various MIS environments.

Protocols: The Hosts

Hosts: HP 3000

Only two protocols will establish a network terminal session with an HP 3000 host. One is AFCP (Avesta Flow Control Protocol). AFCP is the protocol that the DTC uses to communicate directly with the 3000 host. AFCP is not available to the desktop. It is an HP proprietary protocol used exclusively for communication between HP 3000 XL machines and the DTC. Serial inputs or, more recently, Telnet inputs to the DTC are converted to AFCP for communication with the 3000. Telnet is the standard virtual terminal protocol for the TCP/IP transport.

If you want to communicate directly from the desktop to the 3000 over a network link, there is only one option—NS/VT. NS/VT stands for Network Services Virtual Terminal. NS/VT is the equivalent to Telnet, in that it is designed to support virtual terminal communication over the TCP/IP transport. But NS/VT is proprietary to the HP 3000 environment, while the Telnet protocol is supported on virtually every host platform. (One third-party vendor offers a Telnet implementation running on the 3000, but this implementation supports only the "classic" machines.)

Hosts: Unix Hosts, Including the HP 9000, and Other Hosts Supporting TCP/IP

The standard means of creating a terminal session with a UNIX host is through the Telnet protocol. For all UNIX systems (including SUN, DEC ULTRIX, RS 6000, etc.) the TCP/IP protocol suite is standard. TCP/IP includes Telnet, FTP (file transfer protocol) for file transfer and NFS (network file service) for file sharing. All of these protocols are application protocols that run on top of the TCP/IP transport protocol.

TCP/IP is gaining rapid acceptance as the internetworking protocol. Almost all of the major minicomputer and mainframe manufacturers are supporting a TCP/IP implementation. Most noticeable among these is IBM. IBM recently released a barrage of products for the mainframe environment designed solely to enhance TCP/IP connectivity. A standard equivalent to Telnet has been defined for the 3270 datastream (IBM's datastream for mainframe-to-terminal communication) to support virtual terminal over the TCP/IP transport. The standard is known as TN3270 (Telnet 3270). A desktop device with an implementation of the TN3270 standard can communicate directly with an IBM mainframe equipped with TCP/IP, or alternatively a TCP/IP-to-SNA gateway device.

Hosts: Digital VAXs

It is not uncommon to see stand-alone VAX computers or VAX clusters in an HP environment. A fairly large percentage of HP shops need VAX connectivity. This trend is definitely on the upswing as each manufacturer touts the interoperability of its respective platform. To communicate with a VAX from the desktop, you have two choices. The first is LAT (Local Area Transport). LAT is a DEC proprietary protocol that supports virtual terminal communication between a DEC host and a desktop device or terminal server. LAT communication capability is bundled with all VAX hosts. LAT is an efficient protocol for local communication, but because it cannot be routed, it is not used in remote communication.

The other alternative for the VAX is TCP/IP. TCP/IP is purchased as an option for the VAX. It is available from DEC as well as a variety of third-party vendors. TCP/IP on the VAX is very common today and its popularity is definitely increasing because of DEC's failure to deliver a true OSI implementation. With TCP/IP on the VAX, the user can Telnet to a VAX host from his/her desktop.

Protocols: The Network Operating Systems

Communication with the host systems is only half of the picture. The other half is communication with the network operating systems.

Novell

To communicate with the Novell server, the desktop user needs to run the NetWare client on top of the SPX/IPX (Sequenced Packet Exchange/Internet Packet Exchange) transport protocol. The SPX/IPX transport is the native protocol for NetWare servers. Novell has begun development to provide access to NetWare services via TCP/IP.

LAN Manager

LAN Manager in its various flavors (Microsoft, HP, AT&T, Pathworks, Ungermann Bass) is transport-independent. Because LAN Man depends on a NetBIOS transport layer interface, any transport protocol that can support the NetBIOS interface can be used for LAN Manager server communication. Some of the more common are NetBEUI, shipped with Microsoft's and IBM's versions; NBP, shipped with 3Com's implementation; and XNS, originally shipped with Ungermann Bass's LAN Manager. These protocols are efficient for local area communication but lack capability for wide area communication. Recently a number of companies have begun offering a NetBIOS interface to TCP/IP as an optional transport for LAN Manager, primarily because of its wide area and internetworking capabilities.

NFS

While NFS is not officially a network operating system, in practice the NFS standard is commonly used for simple file sharing and often in place of the the "name brand" network operating systems. All NFS implementations require the UDP/IP transport (a subset of the TCP/IP transport). NFS servers are available on many platforms but are most commonly used with UNIX-based hardware. The most popular implementation of NFS is PC-NFS from Sun Microsystems.

Banyan

Banyan uses a combination of its own exclusive protocol and the TCP/IP standards.

Simplifying the Milieu

Outlining the situation is only the first step. Given an understanding of the problem and a good feel for the "linguistics," how does a systems manager go further to enable the communication?

Let's take a look at the three basic strategies:

Homogenize Your Network Communication.

One strategy that many corporations are pursuing is standardizing the language they use for all their network communication. Originally the plan was to adopt the OSI protocol suite, but the lack of significant commercial OSI products has accounted for a very slow adoption of these standards. Instead of OSI, many corporations are standardizing on the TCP/IP protocol suite. Its primary advantages are its widespread availability and its track record.

There are benefits and drawbacks to the single-protocol approach. The benefits are that a single protocol eases network management. The routers, the network management utilities, and even the network engineers have to be concerned with only one type of protocol. Productivity is increased by smoothing learning curves, and even the cost of equipment may fall if the chosen protocol is widely supported.

One drawback to the single protocol strategy is that it is difficult to optimize a single protocol for the variety of communication required on a corporate network. For instance, while TCP/IP is a very reliable transport for a number of communication links, IPX and NetBEUI are optimized for LAN communication. By standardizing on TCP/IP, the network manager may be sacrificing some performance. Another drawback is that despite wide support for a standard protocol, there may be popular (perhaps even required) applications that still do not support the standard. Novell NetWare's current lack of support for the TCP/IP transport is one example of this. Finally, while today we see most of the significant players in the computing world espousing an "open systems" strategy, it difficult to believe that the struggle for market dominance based on proprietary architectures is completely over. If this is true, the network manager may be exposed to some risk by committing all of his/her networking resources down a single path.

The benefits and the drawbacks of the single-protocol approach mirror the ongoing struggle between standard and proprietary approaches to network architecture.

Use Gateways or Terminal Servers to Translate Your Protocols.

Many network managers have used gateways or terminal servers to minimize the linguistic difficulties. Gateways essentially translate specific protocols between disparate senders and receivers. This is done in a variety of ways. The gateway can encapsulate one protocol within another. The Novell server performs this function when it encapsulates IPX packets within TCP/IP datagrams in order to support wide area communication. The NS LAN gateway from HP does the same with IPX and NS/VT.

Some terminal servers can perform essentially the same function when they act as a session management device. For instance, in the DEC world it is common for a LAT terminal server to

advertise a Telnet service (host). The user might use LAT to log in to the terminal server and use the Telnet capability within the server to establish a session to a UNIX host. In the HP world similar functionality is available with the new Telnet Access product, which allows the user to connect to the DTC via Telnet. These inputs are translated to AFCP for communication with the host.

The terminal server's primary advantage is that it can support dumb terminals as well as PC connections. The gateway's primary advantage is that in low-traffic installations it can be used for functions other than protocol translation.

The drawbacks to protocol translation is that it is simply a lot slower than direct connections. The stripping and encapsulating process in the gateway, and the necessity for dual session maintenance in the terminal server, tend to decrease performance. In addition, these devices have a tendency to become bottlenecks in the network and create simply one more hardware node to manage.

Make Your Desktop PC Multilingual.

The final strategy is to give your PC the ability to make connections using a variety of protocols. In the past this was virtually impossible. Network card manufacturers provided only direct drivers for their adapters, and protocol providers wrote their code to interface to these drivers. This meant that a single network card could handle only one protocol at a time. In order to communicate using another protocol, the PC user would have to reboot or install an additional network card. Recently, two standards have emerged to allow a single network card to handle multiple protocol stacks. These standards are NDIS (Network Driver Interface Specification), sponsored by Microsoft and 3Com, and ODI (Open Datalink Interface), sponsored by Novell. The two standards have changed the entire scope of PC communication. Network card manufacturers now supply NDIS and ODI drivers, and the protocol providers write their stacks to comply with these drivers. Now the PC can support multiple protocols with a single network adapter.

A large number of companies today support PC-based TCP/IP stacks that are compliant with the NDIS and ODI standards. Some support both LAT and TCP/IP. Telnet is commonly supported for virtual terminal connections, while NS/VT is supported in a few cases. In addition, almost all of the network operating systems have designed their transports to run over NDIS or ODI. This means that PCs equipped with the compliant protocols can support multiple Telnet, LAT, and NS/VT sessions while maintaining their LAN server connections.

There are significant advantages to the multiprotocol PC. The most obvious one is the PC can communicate with a variety of different hosts and servers without being concerned about which protocol the node is using. This is especially important in larger corporations or universities where users often have to communicate with a variety of different hosts and servers. The communication itself tends to be dramatically faster, because it is limited only by the particular network's

bandwidth and routing capability. Another advantage is that this kind of solution allows network designers flexibility in approaching the whole protocol dilemma. They do not have to buy expensive host upgrades or gateways. Nor do they have to migrate to a new standard immediately. Network designers can phase in the "open systems" standards while maintaining those devices that require the proprietary protocols, allowing use of both old and new technologies concurrently.

Of course, there is some price to pay for this kind of functionality. The biggest one is the PC memory overhead. With all of these protocols loaded, the user will consume a significant portion of conventional DOS memory. This is mitigated by the fact that many of the protocol providers are beginning to supply Windows-compliant protocol stacks that can be installed in extended memory. In addition, many of these suppliers allow the protocols to be unloaded from memory on demand or loaded high with memory managers or DOS 5.0 or higher.

Summary

The objective is clear. From the network engineer's perspective, one wire from the desktop should allow the PC user access to a variety of host and LAN services. This paper has defined the communication problem in the context of the proprietary and "open systems" architectures that define today's networks, and the barriers they present to the potential integrators. It has also laid out some of the options the network engineer or MIS manager may consider in solving these thorny problems.

It is my position that the key factor in deciding which strategy to pursue is flexibility. Flexibility is important because the transition towards standards seems to move in waves, and it is difficult to know whether we are just gathering power or cresting. HP's Network Services, DECnet, and IBM's SNA are still the dominant players within their respective worlds. TCP/IP is showing tremendous promise in terms of availability and applicability, but it is not a panacea. OSI is looming as a potential replacement for TCP/IP. And the NOS vendors seem to be pursuing a variety of strategies. It is difficult to predict the network architecture of the future. The safest, surest course for the present is to position your users to respond to whatever changes the future brings.

Network Planners: Is High-Speed Networking In Your Future?

John B. Selep
Hewlett-Packard
8000 Foothills Blvd
Roseville, CA 95747-6588
916-785-4916

This paper explores the critical requirements driving the need for high-speed networks and how a variety of alternatives meet those needs. 100Base-VG, a potential new local area network standard proposed by Hewlett-Packard and AT&T, is well positioned to meet those needs and provide organizations an easy, inexpensive migration path from the networks they have installed today.

Ever since computers started arriving on desktops, we've hungered for speed. Faster computers. Faster disk drives. Faster printers. All in an effort to access information faster.

Computers *are* faster. Over the past ten years, desktop PC processing power has increased over a hundredfold, from the Intel 8088-based IBM PC of ten years ago to today's Intel 486 and Pentium-based systems. Over the same period, software applications have grown in sophistication and information content to the point that typical information items such as a business letter or presentation graphic require more than 100 times more storage space than items we used ten years ago.

Over that same ten-year period, however, the data transmission speed of Ethernet local area networks has remained constant at 10 megabits per second (10 Mbit/s). As a result, the impact of faster processors transmitting larger and larger data files has been increasing network congestion. Network performance is now becoming a critical bottleneck in a variety of key business application areas. Emerging applications will cause even more strain on networks over the next few years.

Applications Drive the Need for Speed

Three different types of application performance needs drive organizations to consider higher-performance networks. Often, the first symptoms of network strain are user complaints about falling network performance and rising response times. Frequently, these symptoms stem from simple network congestion -- too many users attempting to send too much information and exceeding the overall bandwidth of a 10 Mbit/s network segment. In cases where individual application throughput needs are moderate, organizations can raise their overall network bandwidth through a variety of means, including moving to higher transmission speeds, or by appropriately segmenting their existing network.

In other cases, network performance bottlenecks arise from individual application bandwidth requirements, common in many of today's data-intensive business applications, including database, imaging, desktop publishing, network printing, and computer-aided design. These applications require very large amounts of data to be moved from one location to another (server to client or client to printer) in a single burst. For example, a desktop publishing document with multiple typefaces, a bit-mapped logo, four-color graphics, and other information might consume as much as 10 Megabytes of storage for a single page. On a typical Ethernet or Token-Ring network, it can take as long as twenty seconds to retrieve that one page, depending on network traffic. Retrieving multi-page documents can take a minute or more. On a 100-Mbit/s network, the transfers would take one to two seconds per page, dramatically changing the way users can work with data.

Another emerging application area that will require higher-speed networking in the future is multimedia, particularly in the use of real-time audio and video for video conferencing and interactive video. Unlike most current applications, which simply move entire files from one location to another, these applications require packets of data to be transferred on a regular and routine basis. For example, they might paint a moving image at 30 frames per second or fill in a portion of a voice conversation. These applications cannot afford to have a packet of data delayed or dropped due to a collision or congestion on the network. These application needs are not being met today by existing 10 Mbit/s Ethernet or Token-Ring networks.

Speed Isn't Everything

Increased network performance isn't the only requirement, however. Technologies such as FDDI, for example, have offered 100 Mbit/s speeds for several years. But these technologies haven't caught on in the mainstream marketplace because they haven't addressed other needs, such as low costs and offering an easy migration path from an organization's existing network.

Implementing a network implies a number of costs, including cabling, network adapters for each network client, and a concentrator or hub in the building wiring closet. FDDI is anything but low cost. The electronics required to implement an FDDI network are very expensive; about $1,000 for each client adapter and another $1,0000 to $1,500 for the necessary share of a concentrator or hub. Components for a 10 Mbit/s Ethernet network, in contrast, cost less than $200 per connection. FDDI offers ten times the performance, for more than ten times the cost.

A migration path also takes cost into consideration, both monetary costs and the organizational costs of disruptions and training. With the costs of labor rising and the cost of network components (bridges, hubs, LAN adapters, etc.) falling, a significant percentage of the replacement value of an existing 10Base-T network is in the cabling. If migrating to a high-speed network means replacing cabling, additional costs will be incurred. The cost of installing new cabling can run as high as $500 per desktop for installation alone, especially in older buildings. Most of this cost is in the labor required to pull and terminate the cabling -- the actual cost of the cable is comparatively small. Obviously, any networking technology that lets organizations keep their existing cabling would be preferable to one that required them to install new cabling.

A Wealth of High-Speed Alternatives

Ethernet Switching is a cost-effective technique for segmenting LANs to increase overall network throughput, and is an excellent solution where traffic congestion restricts network performance. By dynamically switching packets between connected LAN segments, Ethernet switching allows simultaneous transmissions among pairs of network segments, increasing overall network bandwidth by two or more times the bandwidth of individual LAN segments. However, since individual LAN segments remain limited to 10 Mbit/s, Ethernet switching offers little for applications that require higher-speed burst throughput, or have time-sensitive multimedia requirements.

Fiber Distributed Data Interface (FDDI) is an established 100 Mbit/s networking standard that uses multimode fiber-optic cabling. However, high purchase costs (more than ten times the cost of 10Base-T) and the need to install expensive cabling hurt FDDI's potential. More recent technologies offering the same speed at much lower costs have relegated FDDI to a role as a high-speed backbone network connecting LANs.

Copper-based FDDI (CDDI) is one approach to leverage the FDDI transmission scheme over lower-cost copper cabling. Operating at 100 Mbit/s, CDDI promised many of the benefits of FDDI at lower electronic component costs and lower cabling costs. However, CDDI is still relatively expensive compared to 10Base-T and requires new higher level (Category 5) unshielded twisted pair (UTP) cabling, an expensive alternative for most organizations.

Asynchronous Transfer Mode (ATM) is a packet-switching technology that bears much similarity to a high-speed telephone switch. Primarily for high-end clients, ATM will support complex multimedia applications, bringing switched, dedicated bandwidth directly to the desktop over existing UTP or fiber optic cabling at rates from 51 Mbit/s to 622 Mbit/s. ATM is also well-suited to backbone implementations providing unique scalability and seamless integration of campus LAN backbones into the wide area network. HP intends to be a leader in this area, actively developing ATM technology and participating strongly in the ATM Forum to promote standards for interoperability.

Fibre Channel, a recent initiative between HP, IBM, and Sun Microsystems, is designed primarily for high-speed mass storage and clustered computing, based on an extremely high-speed/low-delay switching fabric. Using fiber-optic cabling, Fibre Channel will switch dedicated circuits at rates from 266 Mbit/s to multi-gigabits/s. Fibre Channel will also be an optimal choice for connecting high-end workstations to supercomputers.

100Base-VG is Hewlett-Packard's approach for sending Ethernet packet information at 100 Mbit/s over Voice-Grade (Category 3) UTP cabling -- the most widely installed cabling for 10Base-T Ethernet networks. Because 100Base-VG operates on any grade of UTP from Category 3 to Category 5, most users will not have to install any new cabling to take advantage of this technology. Since 100Base-VG can easily coexist with 10Base-T, organizations will be able to upgrade seamlessly from 10Base-T to 100Base-VG as individual user needs dictate. The low costs implicit in 100Base-VG and its easy migration path from 10Base-T promise to make it an attractive alternative for most desktop connections.

A Closer Look at 100Base-VG

Two technologies, Quartet Signaling and the Demand Priority access method, are the fundamental components in 100Base-VG.

Quartet Signaling is the approach 100Base-VG uses to transmit 100 Mbit/s data over four-pair UTP cable. This approach lets 100Base-VG deliver ten times the information while utilizing essentially the same frequencies as 10Base-T.

10Base-T transmits data over one pair of the four-pair cable, using a second pair to detect packet collisions. Quartet Signaling, on the other hand, uses all four pairs to transmit data simultaneously. It also uses a more efficient encoding scheme, called 5B6B NRZ, to transmit twice the number of bits per cycle on each pair than 10Base-T (see Figure 1). Thus, with only a slight increase in frequency, 100Base-VG achieves a full ten-fold increase in transmission speeds over 10Base-T. Because 100Base-VG uses a very similar frequency to 10Base-T, 100Base-VG can use the same cabling as 10Base-T with the same connectors, RJ-45 jacks, and cross connects. This approach allows 100Base-VG to meet U.S. FCC and International CISPR emissions requirements.

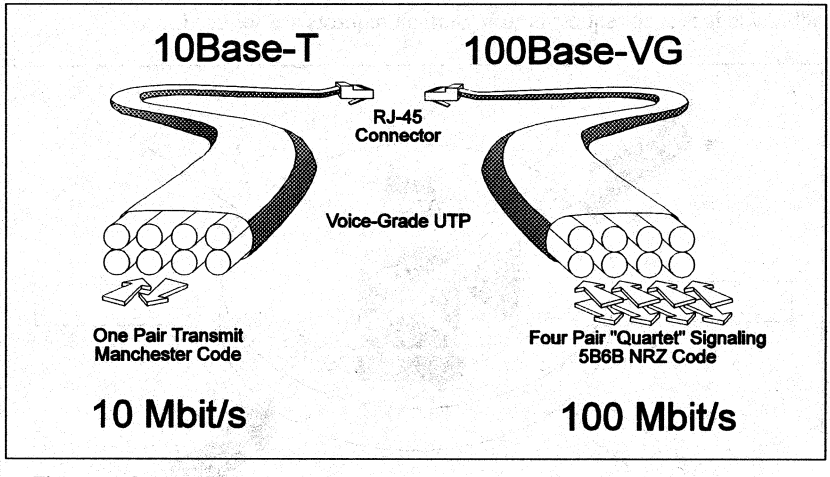


Figure 1: Quartet Signalling delivers 100 Mbit/s transmission over 10Base-T cabling.

Demand Priority is the access method used in 100Base-VG to manage individual devices gaining access to the network. Demand Priority is a simplification of the carrier-sense multiple access with collision detection (CSMA/CD) scheme used in earlier 10 Mbit/s Ethernet networks. By eliminating packet collisions, Demand Priority simplifies network operation and eliminates the overhead of packet collisions and recovery. In doing so, Demand Priority substantially increases usable network throughput and significantly improves network characteristics such as latency, enabling support for time-sensitive applications such as multimedia.

Demand Priority achieves these benefits by taking advantage of the physical topology of today's networks. Most desktop networks are installed using a star topology wiring system, with individual cables radiating from a central hub out to each individual desktop. This includes all 10Base-T Ethernet networks as well as most Token-Ring and even FDDI/CDDI networks. Demand Priority takes advantage of this star topology by using simple intelligence in the hub to arbitrate requests to transmit packets, avoiding packet collisions and significantly improving network control.

With Demand Priority, a node requests permission from the hub to transmit a packet over the network. If the network is idle, the hub acknowledges the request and the station begins transmitting its packet to the hub. As the packet arrives at the hub, the hub decodes the destination address contained in the packet and automatically directs the incoming packet to the outbound destination port (see Figure 2). If more than one request is received at the same time, the hub acknowledges each request in turn, until all requests are serviced.

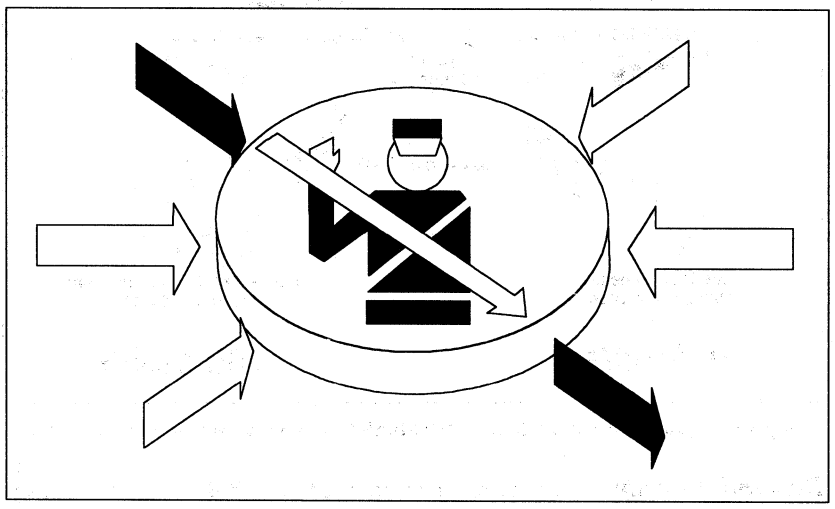


Figure 2: Demand Priority uses the hub to arbitrate packet requests.

Since data packets are directed only to their intended destination port, no other station on the network will see the data packet, its source, or its destination. This provides a level of Link Privacy or security that is not provided today by existing Ethernet, Token Ring, or even FDDI networks.

In addition, since the hub is arbitrating individual requests to transmit packets, the hub can arbitrate different priority requests, acknowledging higher-priority packet

requests before normal priority requests. By knowing the number of applications transmitting at high priority, an application can be assured no more than a minimum delay before its packet will be transmitted to its destination. This effectively provides Guaranteed Bandwidth to these applications, regardless of other traffic on the network. This is a critical characteristic for time-sensitive applications such as multimedia and teleconferencing.

Migrating to 100Base-VG

Migrating to 100Base-VG from 10Base-T is a simple, two-step process. First, the network administrator identifies clients and servers to be upgraded, and replaces the 10Base-T adapter in each workstation with a 10/100 adapter (an adapter that can operate at 10 Mbit/s when connected to a 10Base-T hub and at 100 Mbit/s when connected to a 100Base-VG port). No new cabling needs to be installed. The same RJ-45 connector and cabling used for the 10Base-T network would be used when operating at 100 Mbit/s with 100Base-VG.

The second step is installing a new 100Base-VG hub module in the wiring closet, in parallel with the existing 10Base-T module. Individual workstations could be migrated from the 10Base-T subnet to the new 100Base-VG subnet by simply disconnecting the station's RJ-45 jack from the 10Base-T hub port, and reconnecting it into the 100Base-VG hub port (see Figure 3).

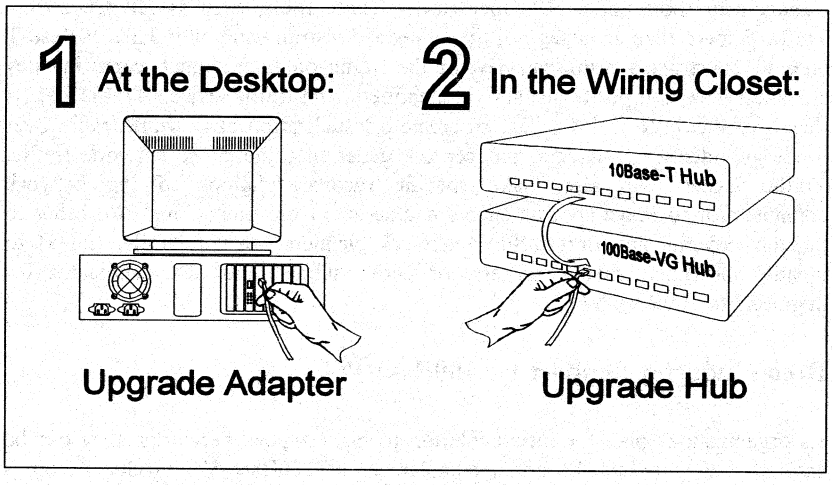


Figure 3: Migrating to 100Base-VG takes two easy steps.

No other changes are necessary in the wiring closet or cabling infrastructure. Individual stations or entire workgroups can be upgraded to 100Base-VG, depending on user requirements. No changes are required for network operating systems, productivity applications, or network management software. 100Base-VG works just as well with the HP OpenView SNMP network management environment as does 10Base-T.

Since the same Ethernet packet format is used on both the 10Base-T subnet and the 100Base-VG subnet, connecting the two subnets into a single network is simply a matter of using a speed-matching bridge to buffer the higher-speed packets as they enter the slower-speed 10Base-T subnet. No packet translation or other processing is required, allowing this function to be an inexpensive component of the hub itself.

You can connect 100Base-VG subnets to existing 10 Mbit/s Ethernet backbones via the same speed-matching bridge approach. Attaching to a 16 Mbit/s Token Ring backbone, using a router, is nearly as easy. Connecting individual 100Base-VG subnets together using an FDDI backbone is relatively straightforward, using encapsulating bridges or routers attach each 100Base-VG subnet to the FDDI ring.

To help users in gaining the best use of their 10Base-T networks and in migrating to 100Base-VG, Hewlett-Packard has released a number of HP EtherTwist traffic management tools under HP OpenView. These tools, such as HP OpenView Traffic Expert, take advantage of the embedded instrumentation in HP's 10Base-T hubs to construct a historical view of the traffic on each subnet, capturing key statistics, including protocols, station identifiers, and traffic levels. Traffic Expert then uses this historical database to create a visualization of network traffic over topology, identifying clients and servers generating the most network traffic. Traffic Expert can also make specific recommendations to the network administrator to move specific clients and servers from one subnet to another to minimize overall network traffic. Network planners can use Traffic Expert to identify individual clients, servers, or entire subnets that are candidates for migrating to 100Base-VG.

Broad Industry Support for 100Base-VG

As organizations plan for their evolution to higher-speed networks, they can be assured that there will be a full complement of 100Base-VG products from a variety of vendors that all interoperate with each other. Even before IEEE standards efforts have been completed, more than a dozen vendors have publicly stated their commitment to work with HP to ensure that their 100Base-VG

products work together. These vendors include leading semiconductor vendors, PC adapter and hub vendors, router vendors, and all three of the leading network operating system vendors.

100Base-VG: The Natural Evolution of 10Base-T

HP believes that 100Base-VG represents the most-attractive networking alternative for organizations looking to migrate to higher-speed solutions. 100Base-VG combines simple, efficient technologies such as Quartet Signaling and Demand Priority while preserving an organization's investments in existing 10Base-T cabling. 100Base-VG will deliver low-cost, high-speed networking with the easiest possible migration path. Organizations can upgrade individuals or workgroups without compromises in cable support, network diameter, network throughput, or cost. Organizations can also take advantage of advanced network capabilities such as built-in security and the ability to support time-sensitive applications such as multimedia.

Hewlett-Packard is a leader in 10Base-T technology today with a complete line of HP EtherTwist PC adapters, hubs, bridges, routers, and HP OpenView network management. HP intends to be a leader in 100 Mbit/s networking technology with 100Base-VG, providing users with an easy migration to high-speed networks supporting their application needs into the next century.

Client/Server with ALLBASE/SQL and IMAGE/SQL

Bryan Carroll, Jim Nagler, and Kriss Rant
Paper 7019
Hewlett Packard
19111 Pruneridge Ave
Cupertino, Ca. 95014
(408) 447-5833

Abstract

ALLBASE/SQL, IMAGE/SQL, and the related connectivity enablers provide the means for third party PC-based decision support and application development tools to access ALLBASE/SQL or IMAGE/SQL databases in a Client/Server environment. This paper will describe the power and ease of use of these tools by presenting simple solutions to complex business problems.

This objective will be accomplished by describing how Graphical User Interfaces (GUIs), Dynamic Data Exchange (DDE), and Multimedia can be used to solve selected business problems. Possible tools that will be described include Forest & Trees by Channel Computing, Impromptu by Cognos, PowerBuilder by PowerSoft, Q+E Database Editor by Pioneer Software, and SQLWindows and Quest by Gupta Technologies. Additional tools not directly addressed in this paper include Visual Basic, Access, Excel and C/C++ by Microsoft and Information Access from HP.

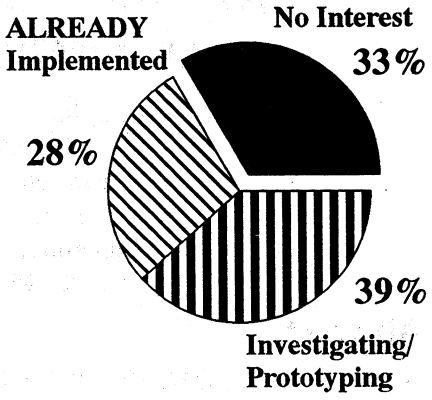
Introduction

A study of Fortune 1000 MIS departments by Boston-based Yankee Group Inc. shows that 69% of the respondents have Client/Server systems or plan to implement them soon - 85% of these use Client/Server for mission-critical applications.

An August 31, 1992 ComputerWorld Survey indicated more than two thirds of those responding have either already implemented a Client/Server solution or are actively prototyping or investigating Client/Server.

The two most common reasons cited for adopting Client/Server architectures were easier database access and to leverage idle computing resources.

Many people are wanting to make use of the PC's they have been buying over the last 10 years for more than spreadsheets and terminal emulators. A very effective use of these cheap MIPS is with a Client/Server application that uses the strength of the PC (Graphical User Interface - GUI) and the powerful features of a relational database.



ComputerWorld: August 31, 1992

Let's begin by looking at what you need to get started with a Client/Server application of your own. Then we'll show how to apply Client/Server technology to some existing business needs.

Client/Server - Required Components

There are a lot of components involved in a Client/Server application, but don't despair, you may already have most of them. Let's take a look at what they are by first taking a closer look at one definition of Client/Server. Client/Server can be defined in many ways, but all definitions involve some 'front-end' system that is mostly responsible for the user interface. We will refer to these as Clients. Another common point involves a 'back-end' system sometimes called a database server, file server or simply a server. We will call these systems Servers in this discussion. The other common point is a link between the Client and the Server or the data communications. Let's take a look at these three areas separately.

Client System Requirements

The Client in a Client/Server application can be almost any input/output device, but the most common ones support Graphical User Interfaces (GUI's). The most common client environments are Microsoft Windows, OS/2, and Unix workstations. We will restrict our discussion to

Microsoft Windows solutions which offers a wider variety of application solutions.

The hardware and software requirements of a Client/Server application in a Microsoft Windows environment are listed in table 1.

1	PC - 386 or better with 4+MB
2	DOS Version 3.3 or later (5.0 or later preferred)
3	Microsoft Windows version 3.0a or later (3.1 preferred)
4	A memory manager
5	HP ALLBASE/SQL PC API
6	Application Software (3GL Windows Development Environment or 4GL Tool or Application)

Table 1 - Client System Requirements

PC

The first requirement defines the PC. Since we are talking about Clients using the Microsoft Windows environment, this must be a Windows capable DOS-based PC. The Client/Server application we are talking about developing will depend on other products like Windows and data communications, so a minimal PC will not provide enough horsepower to provide acceptable performance. Most 386 class PC's will provide sufficient performance. In some cases however, a 486 may provide significantly better performance, especially when considering an Imaging or Multimedia application.

Memory space will also be necessary to provide adequate performance. The actual memory requirement will depend on the environment and required performance, but 4 MB will be sufficient for most environments. Without an appropriate amount of memory, the PC will spend lots of time swapping things to disk. In the worst case, you might need to swap your application to disk while the data communications receives the data from the Server and then swap the application back in (and the data communication out) to process the data. This condition is referred to as thrashing and can easily be solved by adding memory.

As you are probably aware, PC memory comes in three flavors: base, extended and expanded memory. Base memory, also called conventional memory, is the first 640K of memory. Every PC must have some base memory to run and every program that runs on every PC must have at least some minimal space in base memory for the application to run. Some applications require lots of base memory to run, but most will use some base memory and then use expanded or extended for additional memory needs.

Extended memory can be thought of as an extension to base memory that begins immediately following base memory. Windows likes lots of extended memory. Extended memory can prevent Windows and Windows applications from swapping as much. In order to use extended memory you must have a memory manager like QEMM, EMM386 or similar. If you have a choice, you should add extended memory rather than expanded memory to your PC.

Expanded memory can be thought of as a pool of additional memory that can be made available in 16K chunks. Expanded memory is often used by disk caching programs like SmartDrive. You must have an expanded memory manager to make use of expanded memory.

PC Operating System

The next Client system requirement pertains to the operating system which drives the PC hardware. In our Microsoft Windows environment, the operating system is DOS. As with most software, the latest version will contain the most enhancements and features and this is no different with DOS. Version 3.3 will work for most applications, but version 5.0 or later will allow you to use the latest features which include some tuning enhancements. Version 5.0 is generally thought to make better use of memory. Version 5.0 is also preferred when the hard disk is larger than 32 megabytes.

Version 6.0 has not been subjected to a significant amount of testing at the time of this writing, but should provide significant enhancements. Among the expected enhancements are better disk utilization through a compression feature, a memory optimization feature, and improved installation (with an uninstall feature and the ability to install from a network connection).

What we said about version 5.0 of DOS can also be said of version 3.1 of Microsoft Windows. Some very significant enhancements were made to version 3.1 of Windows like the ability to abort a single 'run away' application rather than reboot the entire PC.

PC Memory Manager

We have already mentioned the need for additional memory (beyond the standard 640K). We also discussed the different types of memory. When adding memory, the additional memory will need to be managed by special software for this purpose. Normally managing memory is a function of the operating system, but in the case of the DOS operating system, a memory manager capable of handling extended or expanded memory is not built in. A memory manager such as QEMM from Quarterdeck, EMM386, 386MAX or similar tool will need to be added to your system (unless the DOS version 6.0 features fulfill your needs).

HP ALLBASE/SQL PC API

The HP ALLBASE/SQL PC API product (referred to in the rest of this discussion as simply PC API) is the link between the Client or front-end and the Server or back-end. A collection of dynamic link libraries (DLL's) take application programming interface (API) calls from tools and applications and translate them into requests that go out over the network. The SQL replies are then returned to the calling tool or application. PC API works in the Microsoft Windows environment.

The current version of PC API is based on the Gupta Technologies, Inc. product C/API. The Gupta C/API is a language interface that allows programmers to develop database applications using Structured Query Language (SQL) statements. Currently, the Gupta C/API product is a defacto industry standard that allows tools and 4GL vendors to design their applications to a common interface and rely on the Gupta C/API software to map the calls to one of many different databases. You do not need to purchase the Gupta C/API product in order to use the PC API product.

Not far on the horizon, a more broadly based API is expected to become widely accepted. This new API is based on a SQL Access Group standard. A developers kit was released by Microsoft in 1992. This kit pro-

vides an environment for application and server developers to create interconnections.

This interface is referred to as ODBC - Open DataBase Connectivity interface. The ALLBASE/SQL PC API product will support this new standard. New tools are expected to be written to this interface, and current tools are expected to provide versions that will utilize this interface. At that time, PC API will support both the Gupta C/API standard as well as the new ODBC standard from Microsoft.

Application Software

The last requirement of the Client system is the software that will be your application itself. There are many options here depending on the nature of your development. Some options include purchasing a turn key application, a development environment to develop your own applications in a high level 4GL or a 3GL compiler and associated libraries. We will discuss these alternatives in more detail later in this paper.

Data Communications Requirements

As usual, data communications options represent the largest variables. We will attempt to minimize the confusion by only discussing two popular alternatives, TCP/IP and Novell. The required hardware and software is listed in Table 2.

7	Client Data Communications Software: either HP ARPA 2.1/HP NS 2.1 (or compatible equivalent), or NetWare 3.01b or later for HP3000 Servers.
8	LAN Card for the PC Client
9	LAN/LANIC Card for the Server
10	HP ThinLan/iX (with NetWare/iX if appropriate)

Table 2 - Data Communications Requirements

The data communications requirements will all depend on the particular data communications path you choose. We will be discussing two of the more common alternatives, TCP/IP and Novell. The actual data commu-

nications products that will be required depend on how you are communicating from the Client to the Server. The easiest path for most applications will be to use the PC API product from HP which supports several data communications options.

In its most basic form, PC API uses a Sockets interface. Sockets originated on Unix systems and are often called Berkley Sockets after the University of California at Berkley where they were first developed. When sockets made the move to the PC, each group migrating sockets to the PC environment created their socket package a little different from the others. PC API uses the socket standard known as the HP standard which has recently been adopted by Microsoft and Walker Richer & Quinn, Inc (WRQ).

PC Communications Products

Now that we know we need sockets, what do I buy? Very simply, HP ARPA version 2.1 or HP NS version 2.1 will both provide the necessary sockets software. Future products from Microsoft and WRQ will also have the necessary HP sockets software.

What about Novell? The SPX/IPX protocol can be used with the NetWare product from Novell. Version 3.01b or later will provide the necessary interface (IPX is all that is really used) to work with PC API.

Network Card

The second requirement in this section is the network card for the PC. A variety of cards are available and most will work just fine. The major requirement is that the card be supported by the network software (HP ARPA for example). One consideration, if you will be purchasing a network card, is performance. There are 8 bit, 16 bit and even 32 bit cards available. The faster the card, the better performance can be expected to be. We would encourage you to consider a 16 bit card like HP's 27250B.

A networking card is also required on the server system. If this is an HP3000 system, the HP ThinLan/iX card that comes with the Series 9x7 systems is what is needed. HP9000 systems also have an integrated LAN card.

Server Requirements

This is the easy part! The Server must be an HP system with an HP Relational Database, either ALLBASE/SQL or IMAGE/SQL. There are sev-

eral options available to you, and you can choose the path that makes most sense to you given your current configuration. The Server requirements are listed in Table 3.

1	HP3000 or HP9000 system
2	HP Relational Database - ALLBASE/SQL or IMAGE/SQL

Table 3 - Server Requirements

Server System

The first requirement is for an HP system. Either an HP3000 or an HP9000 can serve as a Server system. Your decision will probably depend on what system you already have available. If you will be purchasing a new system to serve as a Server, you will probably want to consider other activity that may co-exist on this system. You will also want to consider the database you want to use (IMAGE/SQL only runs on an HP3000 for example) and your performance needs.

Relational Database

Which database should you use? ALLBASE/SQL is a full capability relational database with a full complement of supporting products and features including high performance, high availability, interoperability, standards compliance, distributed database, and more. IMAGE/SQL provides a full relational interface to TurboIMAGE databases. This allows many relational features including full relational access (joining tables within IMAGE/SQL and out to other ALLBASE/SQL databases) and full native TurboIMAGE access. The database you choose should depend on your needs, future direction, and where your existing data exists.

Both relational databases provide a similar interface to the data. The difference lies in how the data is stored on disk. Let's take a brief look at the relational interface. You can conceptually think of both ALLBASE/SQL and IMAGE/SQL as sharing a common and mostly equivalent user interface level. Both can parse a SELECT statement, for example. The differences are when the SELECT statement gets translated into lower level storage manager calls to retrieve the data from a buffer or the disk. Figure 1 provides one conceptional view of this architecture.

ALLBASE/SQL-IMAGE/SQL Conceptual View

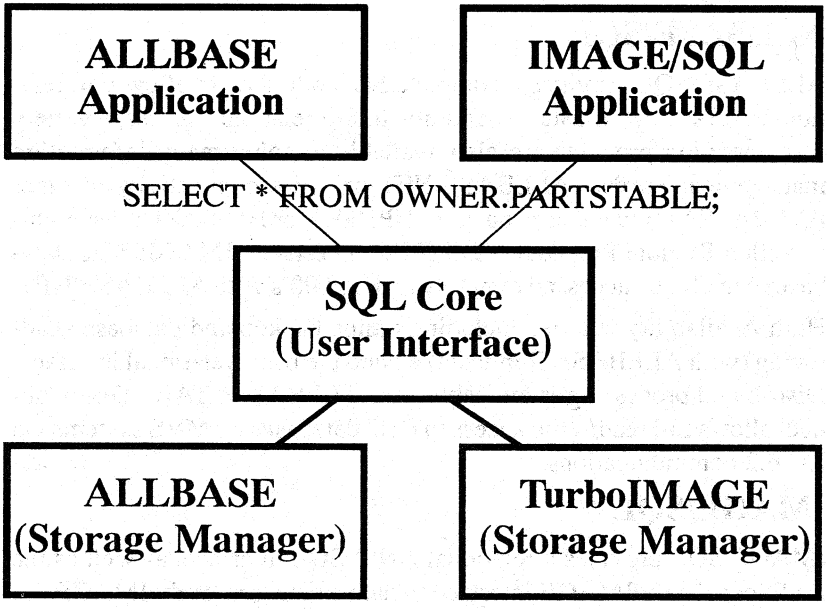


Figure 1

Figure 1 shows that an ALLBASE/SQL application can be coded to perform a SELECT statement. This SELECT statement will be parsed by the SQL Core box. This box represents the relational user interface that understands the SQL language that has been defined by the ANSI SQL Standards committee. After the request has been processed by SQL Core, the necessary calls are made of the ALLBASE Storage Manager, to get the data from the buffers or the disk. This assumes the requested data exists in an ALLBASE/SQL database.

Following the other half of figure 1, an IMAGE/SQL application can be coded with a similar SELECT statement. This SELECT statement would be parsed by the same code in SQL Core that was used to parse the SELECT statement from the ALLBASE application. After processing this request, SQL Core will make the necessary calls of the TurboIMAGE Storage Manager to get the data from the buffers or the disk representing the TurboIMAGE database. Of course, a single application can also ac-

cess both ALLBASE/SQL databases and IMAGE/SQL databases in the same query.

ALLBASE/SQL

ALLBASE/SQL provides a complete SQL interface to allow you relational access to your data. In addition to full relational access, a variety of companion products are also available to solve many information management needs. ALLBASE/NET provides connectivity to other ALLBASE/SQL databases on other HP3000's or HP9000's transparently with a Remote Procedure Call (RPC) interface. IMAGE/SQL databases can also be accessed remotely on HP3000's with ALLBASE/NET.

High Availability features including online backup and database shadowing (with ALLBASE/Replicate) extend the basic relational interface. Distributed processing is available with ALLBASE/STAR. DB2 Connect allows full read/write access to DB2 databases on MVS systems via LU 6.2 communications.

IMAGE/SQL

IMAGE/SQL provides a relational ANSI SQL interface, as well as the traditional TurboIMAGE interface to data existing in TurboIMAGE databases. Full TurboIMAGE access is maintained while allowing relational access for new applications, new application architectures (e.g. Client/Server) or other relational toolsets.

Client/Server Examples

We have been talking about what components it takes to implement a Client/Server application into your company. Let's shift our focus to applying Client/Server technology to some existing business needs.

PowerBuilder Application

This is an example where an entirely new system was being implemented to keep track of customer orders. Orders came in over the phone and clerks would accept the orders which would usually involve multiple line items. This new system was replacing a very old mainframe based batch oriented system. A new database was designed and a new Client/Server application was desired.

In this case, the PowerBuilder development tool was chosen from PowerSoft. PowerBuilder is a powerful object oriented development environ-

ment. Each object can have a unique set of events associated with it and a script can be written for each event. Scripts are written in a powerful proprietary language called PowerScript. The scripting language can handle everything from window sizes and colors to dates and times and even database access.

One example of an object is a Windows Push Button. A Push Button has events associated with it like getfocus, losefocus and of course, clicked. Getfocus and losefocus are Windows events that tell the program when the mouse or pointing device is over the Push Button. You have the ability to write a script when any of these events occur. For example, you might want to change the mouse pointer from the normal arrow to a null sign (circle with line through it) when the Push Button is disabled. The most common event for a Push Button will be the clicked event when the user actually 'pushes' the button.

Database access is primarily handled through special windows called DataWindows. The DataWindow painter made it easy to build the DataWindows and incorporate them into the application. The Menu Painter and Window Painter made it easy to incorporate all the Windows look and feel to the application while making the code easy to develop and maintain.

One special feature that was desired was the ability to encrypt some of the data that was stored in the database. The encryption need, however, was based on the research the company had already done to develop their own special encryption function. This had already been written and put in a library on the PC. With PowerBuilder, it was no problem to include a call to this external library procedure to encrypt the data when necessary.

The entire database was created and maintained from the PowerBuilder development environment. The database painter was used to create tables and views, maintain capabilities and update statistics for the optimizer. Since this was a large project with multiple programmers, the Library painter was used to keep track of the source code and prevent two programmers from making changes to the same modules at the same time. The debugger, with its point and click breakpoints, single stepping and watch window, was especially useful in locating the source of hidden problems.

Another component of the overall design that was added after programming had already started was a link to the existing mainframe. Parts of the old application had been upgraded overtime to exist in a DB2 database on the mainframe. It was determined that in one part of the application, it would be useful to be able to access this DB2 data directly. This need was handled easily with the DB2 Connect product which is part of the ALLBASE/SQL umbrella of products. The NetUtil program was used to tell ALLBASE/SQL about the DB2 database and how to login to the MVS system. Because the DB2 Connect product was built under the ALLBASE/SQL umbrella, access was completely transparent and it was easy to build these functions into the PC Client/Server architecture.

Q+E Spreadsheet Solution

Another business need that arose was quite different. This business already had a working financial application that had been developed over many years. The application stored its data in a TurboIMAGE database and was comprised of host based programs to manipulate the data. Many of these programs required a user to enter or update data while others updated data in large batches.

The problem was the Financial Analyst needed to be able to review this data periodically and make financial projections. These projections were critical to the company's planning. The analyst were currently using hard copy reports together with a spreadsheet (where they entered data manually from the reports) to make their projections.

IMAGE/SQL provides a relational interface to the data in the TurboIMAGE database. Using the HP ALLBASE/SQL PC API product and the Q+E Database Editor product from Pioneer Computing, we can develop a solution where the financial analyst can access the financial data from the spreadsheet on their PC.

Q+E can make use of a relational interface to return data from IMAGE/SQL directly into a spreadsheet on the PC. The Financial Analyst can retrieve the data from the TurboIMAGE database via IMAGE/SQL with Q+E's relational interface. The data is selected from the IMAGE/SQL database as needed using the SELECT statement and all its related functions like SUM and AVG. The data is retrieved from the TurboIMAGE database using IMAGE/SQL and put directly into the spreadsheet.

The data can then be manipulated in the spreadsheet as necessary and the results of the work can be put back into the TurboIMAGE database using IMAGE/SQL. The analysts can extract their data any time they want rather than waiting for the weekly reports to be printed to use for their input. In many cases, the data they were using was over a week old and the analysts feared the data had changed significantly, but they had no way to verify this and update the data they were relying on. The new system allows them to always operate on the most up to date data.

With the write access back to the TurboIMAGE database through IMAGE/SQL, the analysts projections can be put back into the database to be used later for additional projections and reports. The Q+E Client/Server tool has made use of the IMAGE/SQL relational access to get data from the TurboIMAGE database. The analysts did not have to learn a new interface. They continued to use the same spreadsheet they have been using all along which made good use of the inexpensive computing resources on their desktops.

Gupta Imaging Application

Imaging is a very popular technology right now usually involving a Client/Server solution. One example application prototype involved the use of photographs and official documents like police reports. The application required user access by a variety of users to a set of data that consisted of images (photographs and scanned documents) as well as text. This application lent itself very well to the GUI display of the Windows environment as a display tool for the various displays needed for this application.

The large quantities of data fit well into an ALLBASE/SQL database on a server system. Some of the text and financial data, however, already existed in a TurboIMAGE database. This data was accessed with IMAGE/SQL. Both ALLBASE/SQL and IMAGE/SQL were used to hold the data needed by this application.

The application was developed from scratch using the SQLWindows development tool from Gupta. SQLWindows allows you to quickly develop a Windows based application with all the look and feel of windows much like the PowerBuilder product. Development through a working prototype was very fast (just a few days). A full functioning program to capture and store the imaging data was working the first day allowing

proof of concept testing. This was also important since several cameras that translated a photograph image through the camera lense into a bit-map image for storage in the database were to be tested. Revisions to the design were also made early in the process because of the fast feedback madc possible with the quick development cycles of the 4GL.

The main function in the application allowed a picture to be brought up in one window, one or more scanned documents in another window and another window of text and cost information. All of this was coordinated from a parent window where the commands to add, update and delete the data were processed. The application was easy to write in the high level 4GL and the data was easy to obtain even though it resided in several different databases including ALLBASE/SQL and IMAGE/SQL.

This new application allowed the analyst to work entirely differently than ever before where paper and photographs were required. Work could be done much faster and analysis could be done where it made more sense to the company's business, not where a photograph happened to be. For example, if a photograph required some work existed in one state, but the client was in another location across the country, the photograph and other official documents could be viewed by an analyst who was working directly with the customer, rather than across the country by an employee otherwise unrelated to the work.

The reports needed in the application were made easy with a built-in link to the Quest product (also from Gupta). Quest is an end user report writing tool that allows easy retrieval of even image objects fast. Output can be formatted to the screen or go to special devices like a plotter for the pictures. Having the integrated link to a report writer allows the development time to be reduced and it also allows a more flexible end product. Changes can be made more easily, sometimes structuring the program to allow the end user to alter simple things like text formatting, column headings and default values.

Example with Forest & Trees

Yet another example addressed a need for financial reporting in one of HP's manufacturing sites. A financial application system already existed at this site involving lots of code and data stored in TurboIMAGE databases aimed at entering transactions as quickly as possible. Very little

attention had been paid to reporting needs beyond large, often unmanageable, printed reports.

In this environment, changes in business needs, or unanticipated needs often required an analyst to spend several hours to several days assembling data to meet the need. Often, new reports had to be developed to meet the need. This time delay was often a barrier in the decision making process as the time required to assemble the required data.

The solution to this problem was to integrate the data from the 20+ TurboIMAGE databases and make the data available to the end users. A significant part of the solution was to be able to provide a view of the latest data including all up to the minute changes. This is a major step forward provided by this solution over the hard copy report mechanism.

The solution chosen was a Client/Server application using the Forest & Trees product from Channel Computing. Forest & Trees is a Client/Server tool which can access ALLBASE/SQL and IMAGE/SQL from a Microsoft Windows based PC. One of the strengths of this tool is its ability to present data from multiple data sources in either tabular or graphical format.

Color is used to emphasize data that needs special attention because it is out of the expected range. These alarms, as they are called, are especially valuable in this environment where lots of data is evaluated, but the analyst are usually only interested in the exception conditions.

After an exception condition is noted, it is usually desirable to take a closer look at the supporting data to gain an understanding of what caused the number to be out of range. This functionality is provided by a process called 'drill down'. When an exception is noted, the user can click the mouse on previously defined buttons to 'drill down' to lower levels of detail.

Another benefit of the Forest & Trees solution was the time line for implementation. The development team, being used to more traditional development tools, estimated over 6 months to develop and implement the proposed system. The actual time required was much less than half that time. One particular task was estimated to take 2 months and was completed in just 2 days.

Example with Impromptu

The data communications required to implement a Client/Server application are sometimes a barrier due to geography or other considerations. This was the case in another situation where a Client/Server application was desirable, but some of the clients were remote and only had modem access to the server.

In this case, these remote users were technical support engineers who needed to access corporate databases to help determine service contract levels and access a knowledge database to help diagnose problems. The solution was a Client/Server application using the Impromptu tool from Cognos.

Impromptu allows Client/Server access via a Lan or, most important for this situation, via a serial link. Impromptu allows the technical support engineers to dial into the server and issue ad-hoc queries against a variety of databases. A few predetermined queries were saved on the engineers laptop PC in an Impromptu catalog (sometimes called a meta-file). These saved queries could be recalled and modified when needed.

All the power of the SQL language is available to the engineers, but with Impromptu, the engineers do not have to understand SQL syntax. Impromptu can walk you through the creation of SQL statements with menu picks and list boxes. The resulting queries are issued against the server database with the results returned to the PC. Queries can then be further refined to get just the desired data and then the query can be saved in the catalog.

Summary

Client/Server computing can provide very powerful solutions to your business applications needs. There is a strong toolset already available to work with your ALLBASE/SQL and IMAGE/SQL databases. Todays solutions can include imaging, multimedia (including sound), graphics as well as simple text and numeric based applications. You can start with existing data in TurboIMAGE databases using IMAGE/SQL or move your data to an ALLBASE/SQL database.

There are clearly many options and many tools available to help you with your choice. There are many pieces necessary in putting together a Client/Server solution today. We have identified the pieces necessary early

on in this paper, and you likely already have many of the pieces you need to get started. Armed with what you have learned by reading this paper, you should be ready to go out and fill in the gaps along your way to implementing powerful Client/Server applications in your organization.

Trademarks

Authors

Bryan Carroll is currently an Information Management Consultant with Hewlett-Packard in Cupertino, CA where he has worked for the last 9 years. He is responsible for pre- and post-sales consulting specializing in database performance and Client/Server applications.

Jim Nagler has been a developer for the Hewlett-Packard Database Laboratory for the past 5 years. During that time he worked on ALLBASE/SQL PC API, ALLBASE/STAR, DB2 CONNECT and ALLBASE/NET. Prior to these experiences Jim worked as a manager and developer for the IBM/CDC Service Bureau for 23 years on Transaction Processing applications, CALL/370, and DATATEXT.

Kriss Rant has over 10 years of application and tools software engineering experience. He has held various software development and project lead positions on such products as HP Materials Management and HP Purchasing for MPE/iX. Most recently he is a software engineer on HP ALLBASE/SQL PC API, and has been extensively involved assisting various third party PC-based software vendors port their tools to ALLBASE/SQL.

Paper #7020

Troubleshooting Client/Server Applications

Steven L Adams
Information Systems Specialist
Hewlett-Packard Company (M/S 20BBD)
3000 Hanover Street
Palo Alto, Ca 94304
(415) 857-6093

Introduction

In general, troubleshooting requires the ability to ask the right questions. Troubleshooting client/server applications is no different, except there are more questions that require asking. This paper focuses on the key questions that need to be answered while troubleshooting a client / server application.

- Where is the source of the problem?
- What has changed?
- Are standards (if any) being followed?
- How will you escalate problem resolution?

Before the troubleshooting questions are described, the project's infrastructure is listed with comments about the challenges this infrastructure presented to the project team. After which, each key question is characterized and supplemented with examples.

Client / Server Infrastructure of Reference

Our[1] project's client / server infrastructure is important to understand for two important reasons.

1. Our project's infrastructure is different than yours. Client /server infrastructures are like finger prints, unique to an organization and too often unique to a project.

2. By seeing all of the components that contribute to the *whole*, you begin to appreciate why client / server applications

[1] The project being referenced throughout this paper is a Human Resource Management System (HRMS) client / server project. The project is referred to as "our project" throughout this paper.

are harder to troubleshoot than the traditional host / terminal-based applications.

HP's HRMS project infrastructure is as follows:

Client environment (DOS)
- Hardware: HP Vectra, 386 or 486, PCs
- MS Windows 3.1
- LAN Manager 2.1a
- ARPA Services 2.1

File Server environment (LMX)
- Hardware: HP9000 Series 867
- Application Programming Interface (API):
  - Allbase/SQL PC API A.F0.09
- Application executable: PeopleTools 2.11
- Development query product: SQLTalk/Windows 2.0.0
- End user reporting product: NewWave Access A.05

Print Server environment (LMX)
- Hardware: HP9000 Series 745
- Transpooler version 1.3

Database Server environment (MPE/iX)
- Hardware: HP3000 Series 977
- Allbase F.0 (pass 55)
- SQR V2.28c

Network environment
- LAN
  - 10BaseT to the desktop
  - FDDI backbone
  - Hubs
  - Bridges
- WAN
  - Routers
  - InterLAN links (e.g., a T1 leased-line)
- Protocol stack
  - TCP
  - IP
  - IEEE 802.2 (LLC)
  - IEEE 802.3 frames

From this list, you can see there are several components in our client / server infrastructure. However, this does not include the application components! That is, the system components which hold the data and the business rules are not part of the

infrastructure. Therefore, the infrastructure can be used for several applications (e.g., HRMS, accounting, order processing, etc.) and should be[2]! The more homogenous a company can make their client / server infrastructure across applications, the easier and more effective troubleshooting becomes.

Challenges this Infrastructure Presented to Our Project

The foremost challenge presented by our client / server infrastructure was mastering the unknown, which most of the infrastructure components were to our project team. In fact, our alpha-test helped us really understand what our *whole* infrastructure consisted of.

All of the infrastructure components did not require mastering by our project team. There are groups within HP responsible for the networks. Thus, most of the technical details of the networks can be entrusted to these groups by our project team. Likewise, most of the system management details for the file servers, print servers, and database servers can be entrusted to other groups within HP that provide specialization and services to application teams. There is even a group within HP helping application teams by defining client standards and another group providing centralized client support.

So what is the big deal? What in the infrastructure is left? It is true, a great part of our client / server infrastructure is being managed outside of our project team. This is an effective way to build an organization which supports application teams. A client / server infrastructure requires specialization! Our project team is, however, involved with the management of the infrastructure in three significant ways:

(1) 100% of the infrastructure components are not being managed by other groups.
(2) The project team communicates with groups providing these much needed services.
(3) The project team is responsible for answering

---

[2] What organization can afford to develop a separate infrastructure for its business applications? Hewlett-Packard has not done so, nor has any other organization that the author is aware of. This does not mean that every single component listed as part of the infrastructure is shared across all applications. Of course, 100% sharing of the infrastructure is great. It reduces development, training, and support costs. But 100% sharing does not happen very often (if at all). What is being shared across applications are the high purchase-priced items, such as network devices and database servers. The lower purchase-priced items are often not shared across applications. There are some gray areas here and perhaps this is a subject for another paper.

all infrastructure related questions poised by our user community.

What are the infrastructure components that our project team has primary responsibility for?

Allbase/SQL DBMS - our project team was new to relational technology and so was the database server support group. The database support group still does not possess a high level of Allbase expertise compared to Image. (Supplied by Hewlett-Packard)

Allbase/SQL PC API - not only was this product new to our project when we started, it was new to everyone. The product was in development by HP's CSY division. We monitored the product development progress and provided beta-test feedback. (Supplied by Hewlett-Packard)

NewWave Access - end-user reporting tool being recommended by our project to our users. (Supplied by Hewlett-Packard)

MS Windows - we use it and our users use it. We need to know MicroSoft Windows to do our job. Windows was new to the project team, when we started. (Supplied by MicroSoft)

PeopleTools - combination of client executable for the HRMS application and panel development 4GL tool. (Supplied by PeopleSoft Inc.[3])

SQLTalk/Windows - query tool which helps the project team do its job more efficiently. (Supplied to us through PeopleSoft Inc., however, this product is produced by Gupta Inc.[3])

SQR - batch development language which can run on either the client[3] or server -- we use the server-based product almost exclusively. (Supplied to us through PeopleSoft Inc., however, this product is produced by Sybase Inc.; Sybase recently purchased SQR from SQ Software Inc.)

---

[3] Our environment requires this component to run with the Allbase/SQL PC API product; other environments may not have that requirement.

Of course, the best answer to the question, "What are the infrastructure components that our project team has primary responsibility for?", is all of it. Our customers call us when there is a problem. At the very least, for each infrastructure component, our project team must understand the services being provided and the normal operating characteristics of that service. Often our project becomes involved in driving changes in the infrastructure components -- usually to improve performance, reliability, or support.

Four Key Questions when Troubleshooting Client / Server Applications

The following four questions are fundamental to all client / server infrastructures. The previous section described our project's client / server infrastructure for one reason, so you can better follow the accompanying examples provided with each key question. Understanding these four questions will help you develop a troubleshooting strategy. And more importantly, it will help you develop a strategy to reduce troubleshooting events.

Troubleshooting is a complex task under a client / server infrastructure. From the user's perspective, it all looks the same, the application either cannot get started or has stopped working. The user wants to know what is wrong with the application. It may indeed be a problem with the application, or a problem caused by the client, or a problem in the network, or a problem on the server? The first step is to locate the problem source.

Question 1: Where is the source of the problem?

Perhaps the biggest lesson learned during our first year was that when there is a problem, do not assume that every client is affected. Compare this to host/terminal applications: 99% of the time when there is a problem, everybody is down. "Everybody is down" is a bad assumption to use with client / server applications. Problems often affect pockets of users (and if they do not, that is valuable information).

Our project team has defined a downtime-density

metric which reflects this phenomena. The downtime-density metric definition can be found in Appendix A.

Isolate the problem! Find out how many clients are down.

If the answer is one, suspect the client.

If the answer is all, suspect the application, or the database server, or the router connecting the database server with the internet.

If the answer is one site, suspect a site specific LAN, router, or interLAN link problem.

As time goes on, your troubleshooting isolation rules will become a valuable time saver.

Example:

Symptom: user could not connect to a database on Server1, but could connect to a like database on Server2.

The problem was found by isolating the problem. In this case, once the problem was isolated the solution was trivial.

- The symptom eliminates the client as the possible problem, because the same client can connect to the other database (with no changes to the infrastructure components).

- The same reasoning used to eliminate the client, also eliminates the application logic as a possible problem.

- The network components for Server1 and Server2 are the same, which eliminates the network as a possible problem source.

- Therefore, Server1 is where the problem lies. Further isolation tests on Server1, accompanied by some puzzling moments, reveals the problem.

  - First server isolation test: logging onto the server. Result: server1 is up and running. Thus, the easy answer is eliminated.

  - Second server isolation test: logging onto the database through ISQL (a server utility for Allbase). Result: database connection is successful. Conclusion, the database is accessible.

- Third server isolation test: logging onto the account via a terminal. Result: a LOGON UDC (User Defined Command file) with an invalid file equation statement showing itself. Correcting the invalid file equation solves the database connection problem!

Why did correcting the invalid file equation correct the problem? Fact 1: the database server operating system (MPE/iX) tolerates an invalid file equation during LOGON for sessions, but not for jobs (a job will never get started). Couple fact 1 with a second fact, the current implementation of Allbase/SQL PC API creates a job for each client on the database server; the solution becomes understandable.

The most difficult part of the solution was figuring out which server isolation tests to run. It required "What is different here?" answers. That is, the problem was present on Server 1, but not on Server 2.

Question 2: What has changed?

Our project's client /server infrastructure consists of several components, all of which have product version numbers. I suspect yours is the same. Product upgrades have always been a way of life. In a client / server environment, product upgrades happen more frequently There are more of them and product upgrades seem to occur at a continuous rate.

Solving a problem is often found by finding the changed component in the infrastructure.

"It worked yesterday."
"It is not working now!"
"What has changed?[4]"

Identifying a changed component, can be a short-cut to discovering the problem through time-consuming isolation tests.

Hint: Since it is highly likely that something did change, focus your problem isolation activity on

[4] If the answer is, nothing. "Congratulations, you have just witnessed your first miracle." -- a sarcastic quip attributed to Tommy McBride.

finding the changes in the infrastructure (or application) first.

Example:

A database connection problem could be related to a change in the network. A PING test (sending a message to an IP address and receiving a response back) will easily prove or disprove the hypothesis "The problem is in the network" for TCP/IP networks. If the PING test shows no response, then finding a change made in the network can quickly identify the source of the database connection problem.

"Oh yea, the IP gateway on your subnet was reconfigured last night with a new routing protocol." This is what a network support person might say in response to your questioning.

Proactive Steps:

Identifying a change is much more easier when:

1. A list of components that could change is developed (i.e., know your infrastructure!).

2. A detailed change log is kept, which contains the changed component, new version number, date & time of change, and a 'notes' column.

   Problems do not always surface right away. So, instead of "It worked yesterday.", the user may say "It worked two-weeks ago.".

Executing changes under well established procedures will reduce downtime associated with changes -- at least the downtime under your control. Our project developed a change-control model, which has the following elements:

- Approve the change
- Plan the change
- Implement the change in an experimental test environment (and record an entry into the change log).
- Implement the change in a final test environment (and record an entry into the change log).
- Implement the change in the production environment (and record an entry into the change log).
- Review change project, look for improvements

to the process

Question 3: Are standards (if any) being followed?

Implementing organizational standards for the client / server infrastructure is the single most important proactive step an organization can take to reduce troubleshooting, and is the single hardest step to take. The software engineering community typically does not embrace standards fully -- "it is like herding cats". Yet without standards, there is NO client / server infrastructure. There is technology chaos.

Plain and simple: organizational standards reduce the number of questions you need to ask in a client / server environment while troubleshooting. Reduce the technology diversity within the infrastructure components (e.g., network protocol stacks, GUIs, server database systems, etc.) and the more focused your troubleshooting activities will be. With focus, an organization can not only reduce costs by being more efficient, but can be more effective -- thus increasing user satisfaction.

It is relatively easy to standardize on the common components of the infrastructure, as it is to standardize on the high purchase price components of the infrastructure. For example, networking components require standardization or they just do not work (at least not as a single network!). Likewise, it is relatively easy to make sure purchases with high visibility conform to the organization's standards (e.g., database servers). It is much more difficult to standardize on the other components[5]. An organization must plan and work hard to successfully standardize these other components. And it is often impractical to standardize on 100% of these other components. So, focus your standardization efforts on the components that will make the biggest difference in reducing your support costs.

Example:

Within Hewlett-Packard, standardization of the

[5] Such as the components listed in the Client environment and the File Server environment (refer to the "Client / Server Infrastructure of Reference" section found at the beginning of this paper).

client / server infrastructure is in place and is an ever continuing activity. There is one TCP/IP network, two choices for desktop platforms, two choices for database servers, and three choices for database management systems. While not absolutely perfect from a troubleshooting point of view, these standards are a tremendous help because they put significant limits on the deployment of technology throughout the company. The biggest standardization success has been on the PC client, referred to as "HP's PC COE" (Common Operating Environment). The PC is one of the hardest components in the infrastructure to standardize on and it contributes just as much as the other standards in simplifying the environment.

HP's PC COE effort is successful because the project has upper management sponsorship, is well organized, and is staffed with very competent engineers. The PC COE effort has produced several documents, 4 releases (a new release is expected every 6 months), and 100% support from application teams involved with multi-site customers.

What does HP's PC COE standard specify?

- Versions for MS-DOS, LAN Manager, and MS-Windows.
- A common file directory structure on the file server.
- CONFIG.SYS and AUTOEXEC.BAT -- with enough flexibility for customizations, so the scheme actually works.
- Office Automation tools (e.g., a spreadsheet product) -- these office automation tools can be part of the application solution (and are starting to be).

Question 4: How will you escalate problem resolution?

In a client / server environment, the essence of front-line troubleshooting is getting the right people involved with the problem once the problem has been isolated. The client / server infrastructure contains numerous components, so it follows that where you go to get help will also be numerous. Just like it is best to map out a trip before you start, it is also best to know your escalation phone numbers before trouble occurs. For example, it certainly is a frustrating experience to find

out during the pressure of a downed application that a HP Response Center contract needs to be in place before the problem can be solved. Every component in the infrastructure needs an escalation plan.

A HelpDesk is an excellent starting place for the users. A HelpDesk is also an excellent resource for the troubleshooter, both for isolating problems and escalating resolutions. Our group is in the process of developing a multi-project HelpDesk. A data center HelpDesk is already in place, which from our project's view is the first escalation point for most infrastructure problems.

Example:

A particularly difficult problem occurred early in our alpha-test. Our users were able to connect onto the database, but after 5 to 20 minutes the application hung. This pattern happened for the first two working hours, after which most users could not connect to the database. The problem disappeared, unsolved, after lunch. During the morning hours, there were over 20 Service Requests generated by our alpha-test users. Without a reasonable explanation of the problem cause, confidence towards our application and the system infrastructure was nil.

The initial isolation efforts revealed little about the source of the problem. The facts as we knew them during the day is as follows:

- The database server was not the problem. Users at Site 1 were experiencing the problem; users at Site 2 were not -- everything at Site 2 was working fine. Besides, we could log onto the database server and run queries through ISQL.

- The application logic was not the problem, given successful transactions at Site 2.

- The client set-up could not be the problem, both sites had HP's PC COE (internal standard) installed on the clients.

- The networking people at Site 1 reported no problems. (Important to note: the networking people we talked to were responsible for Site 1's LAN.)

What was left? The next day it occurred to us that the problem could be between the two LANs (i.e., WAN) --

although we assumed that Site 1's networking people would advise us of any WAN problems. We found a WAN HelpDesk to call and a reasonable explanation followed. After which, all Service Requests were discarded by our users.

The problem was a faulty leased-line (T1). The HelpDesk reported a leased-line between Site 1 and the database server was shut-off at 12 noon yesterday, fixed in the evening, and was now operational. The HelpDesk also explained the leased-line problem started off slowly (8 - 10 AM irregular client hangs), became worse so no traffic was getting through (10 AM - 12 PM no one was able to connect to the database) until they reconfigured the router and did not use the problem leased-line (after lunch the problems disappeared).

The explanation fit like a glove! If we had known of the WAN Helpdesk number before the problem occurred or had Site 1's networking people known, then we would have saved much worry, saved our user's time (who were very busy documenting the problems), and saved our own time by not exploring the non-problem areas in the infrastructure.

Summary

Troubleshooting, like most things, is most effective and efficient when planned for -- being proactive. The alternative is very costly in time, which translates into excess dollars and missed opportunities. But the most compelling reason to be proactive is to improve customer satisfaction.

Below are the five key tips to help you and your organization troubleshoot in a client /server environment.

- Know (and define) your client / server infrastructure.
- Learn how to isolate problems (and keep learning how to do it better).
- Know what has changed in the infrastructure and when it changed.
- Develop implementation standards within your organization to reduce the technology complexity associated with the infrastructure.
- Know how you will escalate problems depending on where in the infrastructure the problem is occurring.

Appendix A
Downtime Density Definition

A. Purpose
   1. Quantify how much downtime the user community is experiencing. It is essential that analysis of the cause be done. Major causes of downtime (e.g., poor change control procedures) must be identified.
   2. This metric is intended for parties both inside and outside of the project.

B. Scale - hours

C. Definition
   1. Time period reported - calendar month
   2. Hours system is unavailable
      a. Must be within the time window when the system is considered normally available.
      b. Each site will assign a person responsible for coordinating downtime reporting to the metric owner. This data is sent at the end-of-month, no later than 2 business days after the last day of the month. Along with the data, the cause of the problem needs to be noted.
      c. Downtime can also be detected and recorded by the metric owner.
   3. Hours system is normally used
      a. Time of day when users will normally access the system
         (1) Example 1: All sites are in the Pacific Time zone, then the system is expected to be available from 7 AM to 7 PM (PT) for online access.
         (2) Example 2: sites are found in both Eastern Time zone and Pacific Time zone, then the system is expected to available from 4 AM to 7 PM (PT) for online access.
      b. Do not include weekends, holiday, or night shift, unless some of the user community are regularly scheduled to work during those hours.

D. Formulas
   1. Downtime instance = (number of hours the system is unavailable when normally used) * (the estimated number of clients affected)
   2. Downtime density = (sum of all downtime instances occurring within the month) / ((hours system is normally used) * (total number clients))

Paper #7021

Developing Reliable Systems at Minimal Cost - A Case Study[1]

Joe Sitver
Computing and Telecommunications Services
Martin Marietta Energy Systems, Inc.[2]
Oak Ridge K-25 Site
P.O. Box 2003
Oak Ridge, TN 37831-7059
(615) 576-7164

In the pursuit of excellence I present for you here today a case study of a life cycle strategy.

Last year at INTEREX '92, just before Hurricane Andrew touched down, I attended a technical presentation titled "Managing IT/MIS in the 90s". This presentation presented a life cycle theory in the general sense. My aim is to paint a clear picture for you of a life cycle theory through a specific real life project application. I believe that modeling is an excellent way to learn a concept. Having worked in both environments, those that use a life cycle methodology and those that do not, I can empirically compare and contrast the differences between the two.

Creating this life cycle strategy is the first step to achieve a successful project development. Just the action of committing your ideas to paper begins to make them more real. This concept is a tool to concentrate our focus and move us in a direction. Once done proficiently and completely most programming can be done from the plan without having to constantly bother the user. The end result is a concrete contractual agreement between the systems development team and user management who initiated the request.

How are you going to hit a target when you don't even know what it is? With no systems plan the project can be doomed from its conception. There are so many details that go into the design of a computer system. It is inevitable that misunderstandings will occur between the users and developers. If there is no written contract specifying details I can assure you misunderstandings will be extremely costly. I know that there are many environments that operate in this fashion. I have worked in several. Tedious and expensive are words that come to mind. The human brain needs clear, direct signals of what it wants to achieve. That goes for the user as well as the developer. With no common written vision how can we truly understand each other and what we each want to get out of the project?

---

[2]Managed by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy.

Initially the time spent planning a life cycle strategy will be great; however, once the Requirements Document, Functional System Design, Computer System Design, and Impact Analysis are complete the programming efforts will become quick tasks. Precise program test plans tested by the programmer's peer and the user will ensure system reliability.

Perhaps the question is not "What does quality cost?" but "How much does quality save?" For the period 1965 to 1985, Deming Prize winning companies increased sales and profits by 14 percent, 2 percent higher than Japanese companies and 6 percent higher than American companies. [3]

A poorly developed system causes dissatisfaction among our customers. A dissatisfied customer will tell an average of 16 people about their problems. A satisfied customer will tell 8.[4]

This type of life cycle methodology puts you in control of your development project. Listed here are just some of the advantages of this dynamic strategy:

- DEVELOPS RAPPORT WITH THE USERS
- CREATES A TEAM ENVIRONMENT
- INSPIRES CONTINUOUS IMPROVEMENT
- INSPIRES TEAM TO CONTEMPLATE ALL AVAILABLE OPTIONS
- SPECIFIC PLAN
- ELIMINATES UNCERTAINTY
- CONSISTS OF SPECIFIC PHASES TO PERFORM
- CONTAINS MEASUREMENT OF PROGRESS AND STATUS
- COSTS ARE UNDER CONTROL
- SOFTWARE IS EXTREMELY RELIABLE AND QUALITY TESTED
- MAINTAINS FEEDBACK OF RESULTS
- IDENTIFIES TOOLS REQUIRED
- PUTS THE PROJECT IN CONSTANT REVIEW DETERMINING WHETHER TO STOP, CONTINUE, OR CHANGE
- PROMOTES EXCELLENT DOCUMENTATION

The four phases to the life cycle strategy are simply - DDII:

1. DEFINITION
2. DESIGN
3. IMPLEMENTATION
4. IMPROVE

CASE STUDY DESCRIPTION

The case study presented to you is that of a sizeable accounts payable system enhancement. Several

---

[3]Hudiburg, John J., *Winning with Quality, Quality Resources*, a division of the Kraus Organization Limited, 1991.

[4]Deming W. Edwards, *Out of the Crisis*, MIT Press, 1986.

programs will be added and modified in order to incorporate the manual payment records into the HP on-line Accounts Payable system. Accounts Payable Personnel will be responsible for adding and updating payment records and initially creating print files from Lotus 123[5], merging those print files, and uploading the files through Reflection 1[6]. Capability to add cost amounts for miscellaneous, retention, cost, fee, and service date will be added to the already existing manual payments. A cost limits update screen will be created for the purpose of entering cost limit amounts, the period of performance dates, last payment number from the old system, and last cumulative numbers from miscellaneous, retention, cost, and fee. A manual payments inquiry will be created with access to all of the information that currently is available on the Lotus spreadsheet (currently used for reference).

The planning stages involved in this case study were revised many times before the plan was approved. The scope and range of alternative solutions were quite wide. However, as we worked as a team in developing the life cycle (DDII) strategy appreciation of the problems increased. The definition of the system implementation requirements became increasingly explicit. The agreed upon solution was then planned and designed in detail.

Just some of the highlights of what we found out in the planning stages include:

- Several fields were found to be left out of the data base design, screens, and reports
- Path from new data set to an appropriate manual master needed revision
- Some screens had to be reformatted
- Several mathematical edit checks had to be put in place
- Several programs we didn't even think of at the time would be impacted by the scope of this project

If we by passed this critical planning phase the developers would have made some costly assumptions about what the users wanted to get out of this project. Many times these types of changes to the system can have ripple effects. These ripple effects effects cost great sums of money!

I am going to run through my case study trying not to focus too much on the details. I don't want to bother with the details but they are there to show you this is an actual development plan in its authentic form. This will show you the level of detail involved. Specifics are important.

The first page identifies :

- Request Number from the request for automated service
- Document, Design, and Analysis Titles
- Task Description
- Requestor, Requestor's Manager, MIS manager identification, and the dates requested
- Requirements Definition Team which are all the team players including the system development and user team members
- Signature and Dates of both the user representative and project representative. This affirms the acceptance of the terms and conditions within the life cycle (DDII) plan.

---

[5]Lotus 123 is a registered trademark of the Lotus Corporation

[6]Reflection 1 is a registered trademark of the Walker, Richer, Quinn Corporation

SYSTEM DEFINITION -

What is this system required to do? The document begins with the Requirements Document. Here under the heading of Proposed System the vision of the proposed system is explained focusing on what needs to be done to the accounts payable system to satisfy the users' needs.

Planning a system is like planning your childs future. I have a three year old son named Ethan. I have a definition of the kind of man I want him to become. Certain qualities must be attained. He must be loving, educated, healthy, and somewhat religious. So too must we have a requirements definition of a system we wish to create. We need to define the qualities we wish to achieve.

What is the current system? The Current System Definition focuses on the status quo of the users environment. Described are the way the accounts payable department perform their jobs currently. We need a thorough understanding of how things work currently before we attempt to enhance the system.

What exactly is required of the system? Requirements are a picture of what functions the proposed system is to accomplish. Personnel in accounts payable must be able to add and update the several fields via input screens.

What type of output is expected? Output requirements are data outputs in various formats. These outputs are generated from the proposed system for informational purposes. Examples of such outputs include inquiry screens with print capabilities, archival procedures, microfiche, and general reports.

What are the data sources? Data sources define where the source of the initial data. In this case a PC based system is to be replaced and serves as the data source.

Are there specific performance criteria? Performance criteria are specific performance characteristics that must be attained in the proposed system. No duplicate payment records will be allowed. There must be input error checking and data validation as a part of the system.

SYSTEM DESIGN -

What if we could design, define and document the functions of the system all in a level of detail sufficient for the developer to develop the system? Wouldn't that make the whole job easier? The functional system design describes the tasks that must be executed in order to facilitate construction, increase understanding of system functions, simplify testing, and facilitate corrections and modifications. Here a brief description of the current system in place, and the proposed cost type accounts payable system is followed by the business system rules.

To have a degree of certainty that my son Ethan will become a loving, educated, healthy and somewhat religious man I need a plan to get him there. I need a plan to surround him with supportive people, send him to good schools, take him to a religious house of worship regularly, and have him maintain a healthy diet. In short, a plan for life. So too must we have a plan for the life of the system we wish to create. We need a plan to get the system to the qualities we wish to achieve.

What type of people will be supportive of my son? What schools will I send him to? Will his diet be that of a vegetarian or lacto-vegetarian? Which religious house of worship will we attend? A specific approach makes the plan appear true to life. Similarly, planning a system requires planning in specific areas. This will make the system appear true to life.

There are four levels to document the business system rules.

Level one identifies the information coming into the system and going out of the system. Here I focus on the details of type 1 payments and type 2 payments. These are the two types of payment records input by accounts payable personnel. Most of the type 2 payments are cost type, however, only a portion of the type 2 payments are cost type. Identified are the VPLUS screens involved and how those screens will interact in the proposed system.

Level two identifies the information and data flow of daily operations. What type of information do the users create in their work? What information is used by the users to perform work? What are the decision making rules regarding the information? Who has access or receipt of the information? Here these questions are answered. Access to the system at that level will be limited to those individuals who currently make type 1 and/or type 2 payments through the Operator Identification validation process.

Level three details the procedures used with the automated system in its transactions, processes, and data storing requirements. What are the initial log-on requirements by the users? What are the storage requirements of the new data set? Here these are specified. It was determined in this case that a restructuring of the data base would make the data base operate more efficiently if some of the paths were changed between detail data sets and automatic masters. Identified are net gains and losses after each step and the net spaces savings of almost 200,000 sectors.

The inquire print function, by pressing ALT while typing print screen, will provide the user with a screen print on the slave printer.

The upload procedure is a cookbook recipe for uploading Lotus files to the HP3000. Here the steps the user must take are specified.

The data validation procedure identifies the format that each Lotus file must follow. Certain mandatory data edits are identified at the bottom of page seven and top of page eight. The table displays the field name on the left and required width on the right.

The data conversion procedure is where I describe in some detail how the Lotus data will be converted before posting the new data set.

Data set creation explains the new data set, its purpose, structure, and the methodology behind how it will be used. D-PAY-ACT will be created as part of the ACTPAY data base. Both header records and detail records will be stored in this new data set. Certain data will distinguish a header record from a detail record.

First payment on a purchase order function describes how the system will operate in the scenario of a first payment for a particular purchase order.

Subsequent payment on a purchase order function describes how the system will operate for all subsequent payments to a particular purchase order.

The update function is a stand alone program allowing the user to update some important fields that other system functions do not handle.

Inquiry function details the specifics of inquiry. Here all fields displayed, calculations computed, and other features such as a skip to payment number feature are depicted.

Microfiche procedures give some background about how the microfiche records can be found.

Level four describes the system access requirements for the Cost Type Manual Payments System. Who will have access to the system? Only those users with the operator identification that currently can access the Manual Payments - Miscellaneous screen will be able to create these payment records. If entry is attempted with an invalid operator identification that attempter will be locked out of the system.

How is information passed back and forth between programs? Communication between programs and subprograms describe what triggers the calling of the sub program and its functions in specific detail.

What is a data dictionary? The data dictionary identifies in alphabetical order the field names, definitions in user terminology, working storage size definitions, number of bytes, and Image data base structure definitions.

What does the layout for the new data set contain? Layout for the new data set describes the primary key, sort item, all elements, and total bytes required at bottom.

What can the project schedule table do for you? The project schedule table details the steps involved in the life cycle (DDII) stages on the left and provides a method to keep track of estimated start, estimated completion, estimated hours, actual start, actual completion, and actual hours. This table provides high visibility to project schedule status. Total project hours is listed above the table.

What are the processing rules of the system? Here they are laid out. The amounts cannot exceed their limits for all category amounts. Certain edit checks must be performed.

What screen formats are necessary? The screen field requirements consist of tables identifying each VPLUS screen created. These tables describe the field name, length, field type/data type, and processing specifications.

What type of calculations are needed? Specific calculations are laid out. When data is captured, either through an upload procedure or screen entry, several calculations will be made for the entire purchase order so that accurate data will be posted to the new data set.

Is software procurement needed? Software procurement specifications were not necessary here, however, should be described at this point if there are any.

What makes up the system design? The system design identifies both the hardware and software environment. The mainframe to be used for this system is the Hewlett Packard 960. Cobol was the programming language. Several vendor software packages in use include TURBO IMAGE[7] data base, VPLUS[8] screen generator, COBOLXL[9] compiler, ADAGER[10], data base restructurer, and SUPRTOOL[11] data base editor/reporter.

Now the computer system design begins. This design provides a blueprint for the subsequent development of individual programs. General specifications will be produced to define what each program is to do, but not how the program is to be written. Several tables summarize all that was previously discussed into what must be done to make the functional systems design a reality.

- Certain data sets must be restructured
- Certain screens must be created
- Certain screens must be modified
- Certain programs must be created
- Certain programs must be modified

In each of these cases a data set name, screen name, or program name is supplied along with a brief description.

Impact analysis details the impact the systems enhancement will have on other systems. Identified are the task description, impact on specific process, implementation strategy, impact on interfaces to other systems, impact on other systems, inputs, outputs, formsfile to be effected, screen modification requirements, screen creation requirements, report modification requirements, and a table specifying the hours involved for each of the programming tasks involved.

Attachments help to paint a clear picture of what the system will look like when complete. I used several attachments in this case study. While the actual attachments are not included as part of this presentation let me outline them for you here.

- Lotus file layout with fields of interest highlighted
- Inquiry screens show what the uploaded file will look like when viewed with the inquiry function
- manual entry procedure
- Detail payment data is pictured on the appropriate payment screen
- a cookbook recipe identifying the steps of the upload procedure for Lotus payment records
- a recipe to get the Lotus input field sizes in format to be read by the HP Cobol

---

[7]Turbo Image is a registered trademark of the Hewlett Packard Corporation

[8]VPLUS is a registered trademark of the Hewlett Packard Corporation

[9]COBOLXL is a registered trademark of the Hewlett Packard Corporation

[10]Adager is a registered trademark of the Adager Corporation

[11]Suprtool is a registered trademark of the Robelle Corporation

- a payment records data validation report
- how the microfiche program will format the cost type manual payment records
- an ACTPAY data base schematic showing how the data base will look after its restructure
- an actual contract agreement (report) from the current Lotus accounts payable system

IMPLEMENTATION -

How was this system implemented? This system was implemented through the steps of coding, testing, documenting, and installation. A working system that satisfies all functional requirements described in the life cycle (DDII) was developed. The programming was written according to design specifications. The test plan was executed, and the test results were documented. My goal in implementation is the achievement of a smooth, controlled transition from the old user system to the new system which is being installed. Here I developed training procedures and supporting documentation.

In support of the plan to achieve compelling qualities for my dear son Ethan, I must put them in action. I must do what I have made plans to do. Just as I must act on my plan for my sons compelling future, I must act on my plan to create a compelling system.

The last page of this case study exemplifies a test plan format. Proper identifying information is laid out including program name and a general description of the task/action taken. Who is to execute this test plan? Is it the analyst, the user, or both? This will depend on the complexity of the test involved. Instructions for the tester to run the program are included. It is important in the test plan that questions are asked at every step. These questions must be specific and relevant to the test. Questions should be written to elicit either YES or NO answers. Of utmost importance is that the tester be able to run through the test plan quickly. The test plan will contain all signatures at the bottom when complete. In this case where both the user and analyst are active testers, the analyst who wrote the program, analysts peer, user, and project leader all must sign. Typically the project leader's signature signifies approval of the test plan before it is approved for testing.

Efficient, accurate, and complete test plans are to be executed for all programs. As nearly as possible the system was tested under the same kind of daily conditions that would be encountered during regular operations. The objective of the system test is to ensure that risks have been anticipated and that the system can recover.

IMPROVE -

What would happen if each of us decided that everyone does everything for a positive reason? People come to work to succeed. The Japanese concept of Kaizen means continuous, incremental, never ending improvement. By applying Kaizen to all key aspects of work we can all become masters. To become a master, you must always have a 'beginners mind' that is open and receptive to new learnings. In baseball, we have all thrilled at the occasional home run but games are invariably won by singles, doubles, and runs batted in. One great play will not win the game, nor will one great innovation deliver a competitive advantage. Only through reviewing our past experiences and striving to continuously improve on these experiences can we constantly win the game.

To continuously improve my son I will need to study his actions. Things will invariably happen that

are not in my plan. What if I find in his teenage years he is taking illegal drugs? I will need to take immediate action. I will need to interrupt his pattern. Soon he will be put back on the right path. So, too, in the software environment, many unprepared for happenings will come our way. Programming bugs can appear, user environments can change, and platforms can change, but these obstacles need not become stumbling blocks. We must have the sensory acuity to change until we acheive as we desire. Then we will truimph.

Is the production system meeting the original objectives? We must evaluate the system development process. We must look back at the requested system, its development process, the outcome of this development, and perform an evaluation. Benchmarking the most important processes is a great first step. The identification must be made of discrete measurable data. This data should be charted into graphs for evaluation. This evaluation will prove valuable to this specific system as well as to future systems.

Should we be spending our precious time preventing problems or fixing their symptoms? A healthy organization will spend 80 percent of its problem solving effort on preventing problems while a poor performing, reactive organization will spend 90 percent of its time on fixing symptoms instead of causes.[12]

Given a choice, would you rather improve software or merely preserve it? Continuous improvements come from two main sources: innovation and continuous improvement. Current software development processes produce two products: software and defects. The cost of finding and removing these defects is 50 percent of all software costs.[13] This exorbitant cost has become chronic because we have designed the software process to deliver defects as well as software. We must focus on preventing these defects. Quality is a never ending adventure into excellence and mastery.

CONFIGURATION CONTROL

Who controls this life cycle (DDII) process? While it is the task of the developer and project leader to develop the life cycle (DDII) plan it is important that a configuration control designate be in control. That designate reviews the documents for completeness and moves the programs involved from test to production.

DOCUMENT LOCATION

Where do you store these documents? Traceability is a tremendous benefit. To maintain traceability accessibility is required. Documents must be kept in an organized central location. Store all documentation and the latest source code together.

DOCUMENT ADDITIONS

---

[12]Sirkin, Harold and George Stalk, Jr., "Fix the Process not the Problem," *Harvard Business Review*, July-Aug. 1990, pp. 26-33.

[13]DeMarco, Tom, "Software Development: State of the Art vs. State of the Practice," ACM Sigsoft, 1989.

Who is accountable to maintain these documents? It is the responsibility of the developer to attach addendums and/or additional test plans as the life of a system evolves.

A SYSTEM WITHOUT A LIFE CYCLE (DDII)

Now that we have run through my case study example I would like to share with you a system that evolved with no life cycle (DDII) strategy.

Several years ago I hired on to a company in southern California. The lead programmer on this company's cost accounting system, John, was retiring early. My primary task was to learn everything John knew about this system. Documentation, or lack there of, I soon learned was my chief obstacle. Compounding this problem, John had bad feelings toward the company. During his final weeks with the company John would only talk of how awful the company treated him in all his years of service. At all costs he would not divulge the intricacies of his coveted system. Finally John retired. The phone wouldn't stop ringing. Programs were aborting everywhere. Fighting fires became the norm. It was a mess.

- COSTS SPIRALED (IN EXCESS OF $200,000)
- LOSS OF CREDIBILITY AMONG THE USERS
- UNCERTAINTY WAS GREAT
- SOFTWARE WAS UNRELIABLE
- LOW MORALE AMONG TEAM

Obviously this system was like a time bomb waiting to explode! We eventually purchased an accounting system from an outside vendor, tailoring it to our needs.

Have a game plan for the life of your system. While good ideas are important, putting those ideas in writing is imperative. If you don't plant the mental seeds of the results you want, weeds will grow automatically. If we don't consciously direct our minds, our environments may produce undesirable haphazard states. The results can be disastrous.

SUMMARY

The life cycle (DDII) strategy means the ability to change, to adapt, to grow, to evolve, to move in a direction. The life cycle (DDII) strategy does not mean you always succeed or that you never fail. It will give you the power to change the results you're creating. Be a doer. Take charge. Take action. Use what you've learned here. Don't just use it for you - do it for others as well. The gifts from such actions are greater than imagined. To follow this strategy is to produce exactly as you desire. Know your outcome, take action, develop the sensory acuity to know what you are getting and change your strategy until you get what you want.

This case study has given you tools, skills, and ideas that can change your professional life as a Management Information Systems professional. This will help you to become an effective, masterful, elegant developer and communicator. Those with outstanding development and communication skills produce outstanding results. Outstanding results mean the highest quality of software at the lowest cost over the long term.

RAS #6630
MMS REQUIREMENTS DOCUMENT, FUNCTIONAL SYSTEM DESIGN, COMPUTER SYSTEM DESIGN
&
IMPACT ANALYSIS
11/20/92

Task Identification: Cost Type Manual Payments Automation

Requester:	R. Smith	Employee No xxxx	Date 03/06/89
Approval #1:	D.G. Cory	Employee No xxxx	Date 03/07/89
Approval #2:	D.F. Gold	Employee No xxxx	Date 03/21/89
Assigned By:	E.B. Doe	Employee No xxxx	Date 02/08/90

Requirements Definition Team:

Accounts Payable - M. Gold
D.G. Cory

C&TD - K.R. Smith
R.E. Wise
J.A. Sitver

_____ _____
Accounts Payable REPRESENTATIVE DATE

_____ _____
C&TD REPRESENTATIVE DATE

MMS REQUIREMENTS DOCUMENT

PROPOSED SYSTEM

CURRENT SYSTEM DEFINITION

REQUIREMENTS

OUTPUT REQUIREMENTS

DATA SOURCES

PERFORMANCE CRITERIA

FUNCTIONAL SYSTEM DESIGN

BUSINESS SYSTEM RULES

Level 1

Level 2

Level 3

PROCEDURES USED WITH THE AUTOMATED SYSTEM IN ITS TRANSACTIONS, PROCESSES AND DATA STORING REQUIREMENTS:

INITIAL EXECUTION OF THE SYSTEM

STORAGE REQUIREMENTS

RESTRUCTURE OF DATA BASE

UPLOAD PROCEDURE

DATA VALIDATION PROCEDURE

The Lotus files containing the payment records must follow this format standard.

FIELD LABEL	FIELD SIZE
PAY. #	W7
COSTS PAID	W15
FEE PAID	W11

DATA CONVERSION PROCEDURE

DATA SET CREATION

FIRST PAYMENT ON A P.O. FUNCTION

SUBSEQUENT PAYMENT ON A P.O. FUNCTION

UPDATE FUNCTION

INQUIRY FUNCTION

MICROFICHE PROCEDURES

Level 4

COMMUNICATIONS BETWEEN PROGRAMS AND SUBPROGRAMS

DATA DICTIONARY

Field Definitions	Working Storage	Bytes	Data Base Structure
Auditor Number - The operator identification number entered in by the user.	X(3)	3	X10 [1:3]
Cost Amount - The amount of cost dollars to included on the payment record.	S9(9)V99	6	P12
Cost Amount Cumulative - The total amount cumulative to date for the Cost Amount.	S9(9)V99	6	P12
Cost Amount Limit - The upper threshold of cost dollars. The cost amount cannot exceed this limit.	S9(9)V99	6	P12
Entry Date - The date the payment record is entered into the system.	YYMMDD	6	X6
Fee Amount - The amount of fee dollars to be paid.	S9(7)V99	4	J2
Fee Amount Cumulative - The total amount cumulative to date for the Fee Amount.	S9(7)V99	4	J2
Fee Amount Limit - The upper threshold of fee dollars. The fee amount cannot exceed this limit.	S9(7)V99	4	J2
Invoice Number - A vendor assigned reference number.	X(14)	14	X14

Field Definitions	Working Storage	Bytes	Data Base Structure
Miscellaneous Amount - A subset of Cost Amount. This amount cannot exceed Miscellaneous Limit.	S9(7)V99	4	J2
Miscellaneous Amount Cumulative - The total amount cumulative to date for the Miscellaneous Amount.	S9(7)V99	4	J2
Miscellaneous Amount Limit - The upper threshold of miscellaneous dollars. The miscellaneous amount cannot exceed this limit.	S9(7)V99	4	J2
Payment Number - The sequentially generated number used as a unique identifier to differentiate one payment record from another.	X(6)	4 2	X4 X2
Period of Performance (POP) From Date - When the actual service received from the vendor will begin. This date is at the purchase order level.	X(6)	6	X6
Period of Performance (POP) To Date - When the actual service received from the vendor will end. This date is at the purchase order level.	X(6)	6	X6
Purchase Order Number - Contract number. The key element pertaining to the contract.	X(14)	14	X14
Service Date - The last date the service covers on the invoice. This date is at the payment record level.	MMDDYY	6	X6

The following table is a proposed layout of the new data set D-PAY-ACT. All dictionary elements relevant to this system will reside in this new data set.

D-PAY-ACT DATA SET

PURCHASE ORDER NUMBER - PRIMARY KEY	PONO-KEY		
	X(14)		14
PAYMENT NUMBER - SORT ITEM	PAY-NO		
	X(04)		04
PAYMENT LETTER	SUFFIX		
	X(02)		02
MISCELLANEOUS AMOUNT /MISCELLANEOUS LIMIT	MISC-AMT		
	S9(7)V99	J2	04
MISCELLANEOUS AMOUNT CUMULATIVE	MISC-AMT-CUM		
	S9(7)V99	J2	04
COST AMOUNT / COST LIMIT	COST-AMT		
	S9(9)V99	P12	06
COST AMOUNT CUMULATIVE	COST-AMT-CUM		
	S9(9)V99	P12	06
FEE AMOUNT / FEE LIMIT	FEE-AMT		
	S9(7)V99	J2	04
FEE AMOUNT CUMULATIVE	FEE-AMT-CUM		
	S9(7)V99	J2	04
SERVICE DATE / PERIOD OF PERFORMANCE END DATE	POP-DTE		
	X(6)		06
INVOICE NUMBER	INVOICE		
	X(14)		14
ENTRY DATE / PERIOD OF PERFORMANCE START DATE	ENTRY-DTE		
	X(6)		06
TOTAL BYTES = 92	CHAIN POINTERS(8) + 84		

PROJECT SCHEDULE Total Estimated Hours: 542

Manual Cost Type Payment Automation	EST. START	EST. COMPL	EST. HRS.	ACT. START	ACT. COMPL	ACT. HRS.
SYSTEM DEFINITION						
Project Initiation	05/04/92	06/15/92	10	05/05/92	06/15/92	10
Requirements Definition	06/15/92	06/18/92	10	06/15/92	06/18/92	10
SYSTEM DESIGN						
Functional System Design	06/19/92	06/26/92	40	06/20/92	06/26/92	40
Computer System Design	06/27/92	07/13/92	40	06/27/92	07/13/92	40
SYSTEM IMPLEMENTATION						
Programming and Implementation	07/14/92	11/12/92	377	07/14/92	11/12/92	377
Validation and Acceptance	11/04/92	11/25/92	20	11/04/92	11/25/92	20
SYSTEM CONTROL AND REVIEW						
Configuration Control	12/01/92	12/10/92	10	12/01/92	12/10/92	10
Post-Implementation Review	02/15/93	03/01/93	35	02/15/93	03/01/93	35

PROCESSING RULES

Formsfile to be modified: Form:

FIELD NAME	LENGTH	FTYPE/DTYPE	PROCESSING SPECIFICATIONS
MISC	11	R/CHAR	MATCH [-]d+[.dd],[-].dd "Format for Amount is -9999999.99"
SERVDT	6	R/MDY	MATCH dddddd "No slashes or dashes allowed"

CALCULATIONS

SOFTWARE Procurement SPECIFICATIONS

SYSTEM DESIGN

COMPUTER SYSTEM DESIGN

Summary of actions required for task completion:

Data base to restructure: ACTPAY
D-CHECK-REMIT TO D-PO-VEND-XREF, MVEND-XREF - Remove paths
D-PAY-ACT - Add detail data set
M-VEND-XREF2 - Add auto master with paths to D-PO-VEND-XREF, D-CHECK-REMIT Set capacity to 100,003 for M-VEND-XREF2.

Screens to create:
A074 - Inquiry for cost type manual payments

Screens to modify:
A011B - (Type 1 MANUAL PAYMENTS) Add five fields: Miscellaneous Amount, Retention Amount, Cost Amount, Fee Amount, and Service Date.
A133 - Add option for Manual Payments Inquiry (#13)
A008 - (MMS MANUAL PAYMENTS) Add five fields: Miscellaneous Amount, Retention Amount, Cost Amount, Fee Amount, and Service Date.

Programs to create:
MMS1168.SOURCE - Create the validation report of the data file MMS1168.DATA.
MMS1169.SOURCE - Write the data from the uploaded Lotus file to the D-PAY-ACT data set.
MMS1104.SUB - Inquiry for cost type manual payments. Menu driver MMS0392.SOURCE will call MMS1104.SUB.

Programs to modify:
MMS0465.SOURCE - Manual Payments for MMS Orders
MMS0392.SOURCE - Driver program for Miscellaneous Inquiry Selections
MMS0807.SOURCE - Manual Payments for Converted Orders
MMS0521.SOURCE - Archive add
MMS0522.SOURCE - Archive delete
MMS0577.SOURCE - Archive restore
MMS0544.SOURCE - Microfiche creation
MMS0250.SOURCE - AP REFUND CHECK PROCESSING

IMPACT ANALYSIS

Description of Task:

Impact on Specific Processes:

Impact on Data Base Structure:

Impact on Interfaces to Others Systems:

Impact on Other Systems:

Implementation Strategy:

Impact on Interfaces to Other Systems:

Impact on Other Systems:

Inputs:

Outputs:

Formsfile to be affected:

Screen Modification Requirements:

Screen Creation Requirements:

Estimated Hours:

Manual Cost Type Payment Automation	EST. START	EST. COMPL	EST. HRS.	ACT. START	ACT. COMPL	ACT. HRS.
DATA BASE						
ACTPAY	07/14/92	07/22/92	10	07/14/92	07/22/92	10
COBOL PROGRAMS						
MMS1168.SOURCE	07/22/92	07/28/92	50	07/22/92	07/28/92	50
MMS1169.SOURCE	07/28/92	08/01/92	20	08/28/92	08/01/92	20
MMS1104.SUB	08/02/92	08/13/92	40	08/02/92	08/13/92	40
MMS1105.SUB	08/14/92	08/24/92	40	08/14/92	08/24/92	40
MMS1106.SUB	08/24/92	09/14/92	24	08/24/92	09/14/92	24
MMS0392.SOURCE	09/15/92	09/25/92	08	09/15/92	09/25/92	08
MMS0465.SOURCE	09/25/92	10/01/92	16	09/25/92	10/01/92	16
MMS0497.SOURCE	10/02/92	10/08/92	40	10/02/92	10/08/92	16
MMS0509.SUB	10/08/92	10/15/92	40	10/08/92	10/15/92	16
MMS0539.SOURCE	10/16/92	10/30/92	16	10/16/92	10/30/92	16
MMS0572.SOURCE	10/30/92	11/05/92	40	10/30/92	11/05/92	16
MMS0603.SOURCE	11/05/92	11/10/92	16	11/05/92	1/10/92	16
COPYLIB						
COPYLIB3	07/14/92	07/15/92	10	07/14/92	07/15/92	10
FORMSFILE ACCPAY						
A074	07/15/92	07/30/92	4	07/15/92	07/28/92	4
A075	11/10/92	11/11/92	2	11/10/92	11/11/92	2
A011B	11/11/92	11/12/92	1	11/11/92	11/12/92	1

Total Estimated Hours: 377

TEST PLAN

Program Name: MMS1114.SUB

Description of Task/Action Taken:

THIS TEST IS TO BE EXECUTED BY THE ANALYST AND USER

ANALYST: RUN SV50392.JOB.TEST;LIB=P

USER: Select the "ACCRUAL INQUIRY TEST" option in the user test menu.

	Y/N	TEST FUNCTION KEYS
1		Does screen A133, titled "MISCELLANEOUS INQUIRY SELECTIONS", display?
2		Type "14" into the Process Code field and "97YBM424" into the Purchase Order field. Press the ENTER key. Does screen A79 display data for purchase order 97YBM424?
3		Press the F1, Help, key. Does a help screen for A79 display?
4		Press the F2, Print to Slave Printer, key. Is a screen print produced of the current screen contents on the slave printer?
5		Press the F8, Exit, key. Does screen A133, titled "MISCELLANEOUS INQUIRY SELECTIONS", display? Press the F8 key to exit out of the program.

Test is completed; if the user answered NO to any questions then the test has failed.

Analyst: _____

Peer Test: _____

User Test: _____

Project Leader: _____

Paper Number 7022

NETWORKED COMPUTING
"Beyond The Tanglement of Wires"

Richard L. Ponschock, CSP
Revlon Inc.
4301 W. Buckeye Rd.
Phoenix, AZ 85043
(602) 352-5072

INTRODUCTION

This session is intended to provide a management insight to the alternative that a network can provide to the traditional single platform environment. In addition, I will attempt to point out some of the business reasons why a company may wish to downsize to a network based computing solution. The intent of this paper is to arm managers with knowledge of a computer processing alternative that challenges existing mini or mainframe implementations. Cohabitation between mini's, mainframes, and desk top micro's is also viable. However, the current level of development has not provided a utopian environment. Yes! LAN's do still have a few problems, but then so do mini's and mainframes. Some of these short comings will be pointed out.

The design and implementation of a successful network whether a LAN, WAN or some combination of the two is more than just the physical connection of hardware components over a selected set of media. The design must start with a dream, a reason. Unless you happen to be employed by a research firm or have access to a bottomless pocket, that vision is probably driven by a company problem that needs a solution. The business problem could be:

- The need for growth beyond your current Computer
- The need to increase productivity
- The need to decrease the time of information processing
- The need to provide remote retrieval of production data
- The need to reduce or eliminate central processing demands
- The need to tie together departmental developed information "Islands"

The list is as endless as the number of businesses and each of their unique requirements. This address is a presentation of my opinions. Not all of these thoughts are a "household" concept.

In addition, many of you have never heard of Dick Ponschock before. Why then, should you believe anything this "stranger" says? Well, I have had similar thoughts while attending conferences or reading papers like this. Therefore, I have taken a slightly unique approach to substantiate my comments. I have researched and collected quotes from various authors, consultants, I.S. managers and end-users. These fore-bearers add a degree of credibility and credence to this stranger. **NETWORKED COMPUTING**, is a study in connectivity, information and business communications. The subject is made up of both a highly technical component and one of a business impact. I believe that Steve Jobs coined the phrase "The Network is the computer". Today's discussion will go into a good level of detail on how the linking of computer resources, existing or new, can form a singular information platform.

Beyond the technical aspects, is the impact which this architecture has on the business. Some of the issues that will receive explanation are:

- Cost
- Organization
- Rapid change (obsolete decisions)

Finally, the future direction of **NETWORKED COMPUTING** will be examined. We will look at the growth being experienced in the server market as well as the projections for the technology itself.

CLEARING THE FOG
(Defining the "Buzz" words and the "Fuzz" words)

The study of networking information systems on an enterprise level is complex and can be confusing to the non-technical manager. As I was digging through notes and preparing for this talk I re-read the material from a speech I gave to a monthly meeting of the Association for Systems Management just two years ago. Then the "BUZZ" word was "Connectivity". The word "Connectivity" is defined as:

CONNECTIVITY

"The capacity
of the electronic medium
to assist management
in keeping a business organized.
In this sense, the missions
of management, organization
and communications are identical.
After all,
management processes are primarily
a communications activity
for preventing the forces of entropy
and disorder from taking over."

Paul Strassmann

Since that time, many more words and phrases have been added to the list. Is there really a difference between the following vogue phrases splashed around in today's press?

- Networked computing
- Desk top computing
- Cooperative processing
- Host based system/Distributed computing
- Client/Server architecture

If there is, I will not spend time on describing the architectural differences. However, a short explanation of the general concept of this technology will be good, to set the stage for the rest of the session. **NETWORKED COMPUTING** treats diverse processors (PC's, Mini's, etc.) as indistinguishable units on the network. Although the concepts that I presented a few years ago hold true today and in many cases they are still "FUZZY". The concept of decentralized computing power with enterprise wide information access continues to be in a maturation phase of existence.

Some of the terms thrown around back in '89 and even today basically mean the same thing. I think the following does a pretty good job of describing the phrases used to create industry hype.

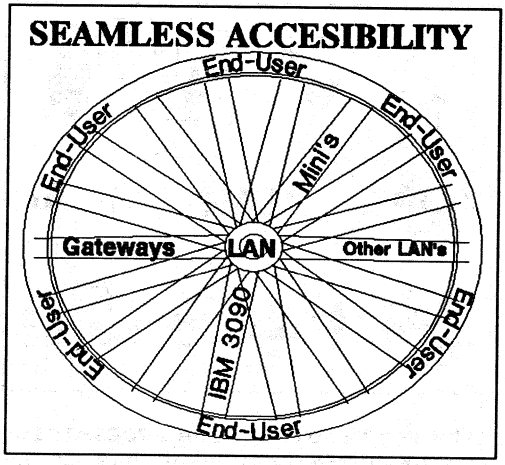
"The phrase "distributed computing" itself is not so much a buzzword as a lint ball: and amorphous collection of fuzzily defined terms, concepts and computing architectures, including PC-to-host, peer-to-peer, client/server, cooperative processing, coherent computing and integrated network computing"

A.E. Alter
CIO Magazine,May 1991

Remember, this is a management overview. As I promised, I will not bore you by describing the technical differences between cooperative processing and network computing or the laundry list of synonyms. Earlier, I indicated that **NETWORKED COMPUTING** was not a "household" word. Looking at today's statistics, the "experts" quoted in the slides, as well as myself, this continues to be true. In 1990 only 15 to 20% of all personal computers were networked. However, these figures were to grow to 40% this year. Maybe stock in a networking company would have been a good investment.

I did promise not to bore you by describing all of the differences between the variations of **NETWORKED COMPUTING** and that promise I will keep. I will however, attempt to give you a broad brush primer on the overall concept. I will introduce Dick Ponschock's explanation to this "high tech" web of computing. In my opinion, **SEAMLESS ACCESSIBILITY** is a better way of describing the **NETWORKED COMPUTING** concept. Maybe, just maybe, I can erase a little of the fog. Let me compare this "tanglement of wires" to something we can all identify with - the WHEEL. Call it a model, an architecture or just a common sense explanation. If we cut through the flowery words, the vendor hype and the theoretical press, I maintain the subject can be boiled down to something understandable by all levels of the organization -the WHEEL.

The Wheel, at least the original design was made of a hub, a rim, and a series of spokes. The spokes were inserted to keep the rim from collapsing and being useless. I call this my **SEAMLESS ACCESSIBILITY** model. **SEAMLESS** because the users, regardless of where they may be in the enterprise, have equal **ACCESS** to all informa-



tion and the paths to that information. I must point out that this model assumes that the ring provides routing from service to service. A spoke provides ACCESS to an application or is a service by itself. The hub in my **SEAMLESS ACCESSIBILITY** model is a Local Area Network (LAN). It may not be the only LAN as the diagram shows. The LAN in today's technology development is the most OPEN framework for connecting uncommon or heterogeneous computer systems. If we need to connect a part that we didn't know about, just add another spoke. That spoke may be a router, or a bridge to another information source. For those managers that have no idea what a router or a bridge is, I just wanted to include a little computer jargon for the benefit of our "techy' readers.

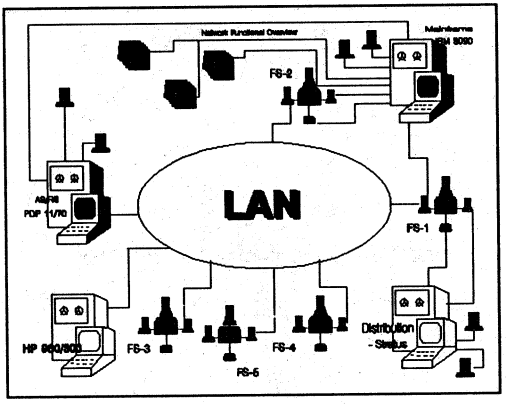
I just mentioned a very important and key point to the networked computing **SEAMLESS ACCESSIBILITY** model - OPEN framework. What if we had to access a technology that we didn't even know about at the time the network was installed (ie. Optical Disk, an analog interface to a production machine, etc.). The **SEAMLESS ACCESSIBILITY** model has the ability to deal with that demand.

Well, how does this model look in real life. The diagram shows the Revlon Phoenix network.

Let me contrast this to the WHEEL. The LAN is the cohesive attribute, the infrastructure. It is the foundation to which the other components can attach.

- HP 980/300
- DEC
- Six LANS
- Mainframe
- Stratus

If we sift through all of the technology issues, the main concern that you as managers need is the assurance that the model has a way to get information from any computer platform that your company is now using. This access should be able to take place without having to know that one or more computers are involved. I believe that we are approaching that capability very rapidly. The **ACCESSIBILITY** is now available. The **SEAMLESS** component has to be fine tuned a few degrees.



Ok! If you quit reading now, you will think that building a network and network strategy is pretty clean and problem free. Those of you with battle scars are asking yourself and maybe a colleague "What is this guy smoking?" Well, building the **SEAMLESS ACCESSIBILITY** model is possible. It is not painless and not all "Milk and Honey" Craig Burton in an article in June's issue of NETWORK COMPUTING draws an excellent analogy.

> "When I think about the issues involved in connecting LAN's to mini's and mainframes, they seem similar to the issues facing the Soviet union as it changes to a free economy. It's surely the right idea, but getting from here to there is a real bitch."

We must be able to access, store, and move files effortlessly between host, LAN's and workstations on the LAN's. All of these can be done today, but not in a painless, effortless, and **SEAMLESS** environment.

BUSINESS ISSUES

Technology issues are not the only problems facing the networked computing world. As managers, each of us must look beyond the technical aspects of a new idea. There are many concerns that we must deal with during the

period of transition between centralized and desk top owned information. Time will only permit exploration of three of these management issues:

- Organizational structure
- Capacity
- Information Sharing

ORGANIZATION

NETWORKED COMPUTING is something of a contra-distinction. While it puts data and applications close to those who need information and computing power, controlling the distributed computing environment requires a centralized mainframe mentality. I debate this issue with myself on a regular basis. How can network support, and control be maintained by a central I.S. function without building the empire of the past. This is not easily answered. I am not going to pretend to have a solution for everyone.

The well defined organizational structures that existed between I.S. and user departments dissolve as distributed computing becomes more ingrained in the company. Today, CIO's need to be in control of the computing power, but they can not be autocratic or arbitrary about the way they manage it. Information executives are creating solutions to this situation by sharing power while finding ways to maintain and increase central control of the computing backbone. The rigid bureaucracies and lateral organizational relationships do not hold up under flexible distributed technologies. How does I.S. and today's CIOs restrict harm from conflicting applications, overlapping data bases and technical incompatibility as users control their own software development and in many cases the entire departmental domain?

Top management must enforce the principle that information belongs to the company and not to a singular department or function. Back in 1988, Forester Research Inc. predicted that by 1992 40% of applications will be created outside of I.S. influence. In a September 1991 article, Joseph E. Izzo, the V.P. of A.T. Kearny a Chicago based management consulting firm predicts that 70% of software development will be done by end-users. For the users in the audience, there is a silent applause. However, for I.S. management a cold chill is progressing up the spine. The fear is not a loss of an empire or dynasty but how is the unknown, unmonitored,

and uncontrolled explosion of dispersed development going to be supported as these "one man" development teams leave the organization, get promoted or dig in over their heads.

As almost a natural process, for each departmental work group or server a power user seems to surface. These individuals should be harnessed. They can be a valuable extension to the conventional I.S. department. Most of these power users are writing programs in various languages for their department and establishing systems that may or may not fit the corporate objectives. I suggest that these individuals be formally included into many of the goal setting sessions of the central I.S. organization. They can be a valuable asset if directed properly. Establish a management sanctioned review group made up of the power users from each department. This group, will begin cross fertilization. The first mission is to increase the awareness and:

- The exposure each department is contributing to the company.
- Inform each other of available information from central data bases or other departmental systems.
- Provide a forum for bringing the personal applications "out of the closet".
- Begin pre-approval of future development to ensure the current data bases have been leveraged to the maximum.

I believe I.S. must also take the leadership roll of training the end-user community, particularly the "power users". The "power users" can the first group addressed in the education process. Take note, education can not be limited to the users. Traditional mainframe I.S. must also be retrained. They must be enlightened in the PC world. The traditional developers must first understand and appreciate that the PC is not a "toy". PC's are very serious players in the information technology arena.

I.S. management must also show leadership in expressing and promoting joint decision making in design of applications. This joint decision making cannot be misconstrued as a ploy to regain territory. The client/server revolution is fueled by the need for American companies to become more responsive. Business today must be more flexible. Applications must be able to change and be

changed rapidly. The pace of change has moved past drive one and into overdrive. The company that will survive must adapt quickly to change. The reaction time of the traditional mainframe system development life cycle will not be acceptable in the '90's.

With that said, I.S. management must not expropriate its responsibility of enforcing security, copyright compliance, data integrity, adherence to standards and technology consultation. Throw away the old job descriptions they no longer apply.

- End-user must become programmers
- I.S. developers must learn PC's, LAN's and Networks
- CFO's and CEO's must make certain that the new rolls are being embraced.
- I.S. management must take a leadership positions in training, architecture and data cohesiveness.

CAPACITY

The next issue that management must wrestle with is computer capacity. Computer capacity is a major concern of everyone who has the responsibility over information resources. As a member of the company's management team, our commission is to keep the organization out of trouble. Therefore, we must be cognizant of the fact that every computer resource can run out of steam. What happens if the mini that you manage reaches an unacceptable level of response time - UPGRADE? What if you are at the end of the product line - CONVERT? What if the cost of converting is prohibitive? You have or will someday face this dilemma.

In the summer of 1982, I faced this problem. At that time I made a promise to myself. I committed never to put myself in the situation again. I was managing an I.S. Department that utilized an IBM System 34 minicomputer. To make a long story short the system "hit the wall." No I don't mean that someone shoved it into the side of the room. How many of you are joggers or have watched the BOSTON marathon on Television. "Hitting the wall" simply means no matter what that runner tries he/she is out of energy. No matter how hard he/she wishes or pushes; moving another inch is impossible.

Total exhaustion has over taken the body. The mind over matter principle is voided.

This is a familiar situation for those of you who have been involved with pushing a mini-computer to its maximum. When "hitting the wall" occurs and the mini has been expanded to its full capacity what do we do? What if the vendor doesn't have a a larger machine. You have just put the company and yourself in a corner. Both you and your company will suffer until the vendor, on his time frame, releases a new piece of hardware. Well, with today's technologies I don't believe that your destiny and your company's survival has to be put into the hands of another company. Properly structuring a **NETWORKED COMPUTING** environment, you can add to the network and not be limited by the single source of computer power. This approach leverages current assets while protecting your ability to grow in the future. Remember the **Wheel?** Another spoke can be added. Downsize one application. Add a homogeneous low end mini or another server.

INFORMATION SHARING

The third concern that I will dwell on for a moment is information sharing. Sharing, control and integrity of the information available throughout the enterprise is vital to the health of an organization. As PC networking increases, firms are finding a strong need to regain control of their information. In the early '80's many companies began to store tremendous amounts of vital strategic, tactical and operational information. The problem is that it was widely disbursed on PC's or departmental systems which were not accessible to the rest of the corporation. Sure some resourceful users have resorted to "sneakerware". But this approach was far from enterprise wide and further from productive. The "sneakerware" approach was usually only peer-to-peer. Information on a mini or mainframe was not commonly accessible outside of their domains. With the technologies of the '90s, the trend and capability is to pull this information together while still allowing the knowledge worker the independence at the department, work group and desk top levels. With the **SEAMLESS ACCESSIBILITY** model, the knowledge worker can retrieve information from the "Islands" of information otherwise secluded. Data redundancy can subsequently be reduced.

Whether I want to admit it or not, the mainframe manufactures really started the whole business about distributed processing. In order to reduce some of the impact of hundreds or thousands of terminals hammering at the CPU at one time, or reports all being printed in a central location, they began to add "spoolers", dedicated report handlers or terminal controllers and front end communication devices to the systems. They started the migration by distributing some functionality off the central processor. Then came the PC. It gained popularity far above anything IBM or APPLE ever dreamed of. This revolution distributed the information, the terminal and the computing power off the mainframe and mini and put all facets of data processing on the desk top.

> "Movement from workgroup LANS to enterprise LANs is becoming an urgent corporate priority. In the '80s, the workgroup was typically automated. Now, workgroups must be integrated at both the division and enterprise levels"
>
> Phil Paul
> Peopleware July 1991

Integration without the loss of departmental and desk top freedoms must be the marching orders of today's system designers and business systems integrators.

TECHNICAL CHALLENGES
(Alternatives, Trade Offs, and Pitfalls)

Using the newly coined phrase **SEAMLESS ACCESSIBILITY** as our corner stone, how close to utopia are we? As discussed at great length and imbedded in my coined phrase is the fact that we are connecting, interfacing and integrating applications, information and the platforms of many generations, vendors and implementation strategies. After throwing this diverse set of technologies, data and visions into a pot, stirring it gently and simmering for just the right amount of time we hope it will turn into the right solution for our organization. How can the risk be removed. No one wants to stick his/her neck out if it can be prevented.

The safest approach is achievement by evaluation and modification. Start by connecting your PC's or current

departmental network to your existing mini. The movement will carry itself from that point.

> "No system will be perfect in its initial design. In the same way no model or process should be considered fixed. As needs become more clearly understood, and as technology and products evolve, the organization's model will evolve. "
>
> Steve Malisewski
> Microage Quarterly
> Vol 3, no 2

Those of us even remotely involved with the management of information technology are in both an enviable and fearful position. Any decision made can and probably will be second guessed in a year or two. If you chose Ethernet (a type of wiring topology) three years ago, you could be asked, today, why are we not using token ring or 10 base T? If you programmed an application in dBase, why not Paradox? If you chose Multi-Mate as a word processing standard, why didnt you select Word Perfect?

Well, if you settled on NOVELL you are probably on safe ground. Let's for the sake of argument say you chose:

Word Perfect
Lotus 123
Novell
dBase
Token Ring
TCP/IP

Some would say you hit six out of six. Not bad! Well don't pat yourself on the back too much. You will soon be questioned:

- How come we are not running Word perfect 5.0?
- DOS 6 is out, we need the memory manager feature.
- Why doesn't our version of LOTUS link spread sheets?

At the same time your management is getting a little gun shy because of continual requests for funding a new release of software every couple of months. Upgrading is not a nickel and dime proposition. Those of you who have just contemplated a change from NOVELL 2.xx to 3.11 know

it is not petty cash. However, you are caught between a rock and a hard spot. If you elect not to upgrade, you contend with a support issue or miss a new feature that is needed by some users.

The industry is in a rapid maturation period. The PC is only ten years old. Imagine the explosive changes that have occurred in that period. Also keep in mind that 95% of all PC's used in corporations have been purchased since 1983. I only have a few suggestions:

- Because of the rapid software releases, don't be tempted to violate Copyright licenses to save money.
- Budget for some upgrades every year.
- Don't feel compelled to rush into a new release unless you have bugs in your current one or need a new feature.
- In the area of operating systems and Data Base managers, I would not jump into a x.o release. I suggest waiting for a sub-release.

As you weigh the current trade offs in both hardware, and software, make an educated decision at the time you need to solve the business issue or problem facing you. Don't second guess your decision. If what you have installed is working, don't change because of pressures from vendors or the press. There will continually be better "mouse traps". In an industry that is changing as rapidly as the one we are asked to support, there will be something better probably before you get yours installed.

On the other hand, the OPEN nature of the PC and LAN technology will allow you to change if you did make a error in judgement, your business requirements have shifted or there is just a better approach available today. In most cases the majority of your initial investment is probably still intact. The most radical decision is probably faced by 3-COM users. However, even they can convert to NOVELL and retain the PC workstations that have been put into place. Most word processors have the ability to translate to other leading packages and the major spread sheet programs can import and export data or even read the competitor's files.

Even your connectivity strategy can be questioned. Emulation cards were the best thing since sliced bread a

few years ago. They can now be viewed as a mistake and a gateway is the "right" way to bridge to a mainframe.

> "Emulation cards provide a simple way to connect the mainframe to the PC"
>
> Kurt Christoff

Well, if what you have is working, pat yourself on the back. If the approach isn't adequate, migrate. There are many options for each technology issue and more options available everyday. Continuing to attend meetings like the one we are at will make you aware of your options. Remember; if you keep waiting for next year's car to come out, you will continue to walk.

IT's LONELY STANDING BY YOURSELF
(How's the company we keep)

Sometimes there is comfort in numbers. I have also heard that misery loves company. Downsizing applications to micro based technology can be and is viewed by some as a risky venture. Mainframes still view PC's as non important even "toys". Well, let's look at those who have gone before us and determine:

- What business problem these pioneers solved.
- What approach they used as a solution.
- The benefits that their company achieved.

One reason cited for downsizing is cost. The I.S. director of New England Energy Company stated in a Computerworld article entitled The Dynamics of Downsizing way back in '86 that he was then paying $700,000 a year to maintain their mainframe and $80,000 a year just for the terminals. A $ 94,000 expenditure would replace the terminal costs. Not bad, **but** not outstanding. A little over a year payback. However, the bonus was, that for about $200,000 more, they could replace the mainframe maintenance costs. These reductions were based on downsizing to a network solution. We can't forget how PC prices have dropped since '86.

Although I deeply believe in the "smaller is better" and "decentralized" environment, proceed with a degree of caution. Although the benefits of downsizing arise from

a cost orientation and the cost reductions scenarios are possible because of low cost hardware, be certain that viable software exists for the application that you are proposing. I recommend that you search for a commercial system first. If a commercial package is not available, assess the true costs associated with development.

Partial downsizing or incremental downsizing is a safe alternative. By the way, it fits very nicely into my **SEAMLESS ACCESSIBILITY** model. Start by accessing information from the existing application then slowly migrate the entire application to the micro based client/server. Some applications are easily portable. System/36 users can move to a Compaq System Pro with a migration tool that ports the RPG source code to the server. Not all porting is that easy. If the move can be made with relative ease cost advantages await. A 25 users system is about 37% less on a Compaq then on a comparable mini. A $100,000 mini price tag is compared to a micro super server of $65,000. Cost need not be the only motivation. I have chosen a few companies that downsized or networked for other reasons.

CASE STUDIES

ORGANIZATION: Oregon Department of Environment Quality, Portland Oregon

PROBLEM: To expand an enterprise wide access to information and computing services.

SOLUTION: Bridged multiple PC LANS. Distributed networks at each remote office. An additional large LAN at the home office.

BENEFITS: Ability for remote offices to access data bases on headquarters mainframes in addition to local processing. Remote connections support environmental quality.

Connect Magazine, Spring 1991
Aubrey Wallace

ORGANIZATION: Craver Mathews Smith and Company

PROBLEM: Needed a way to maintain mailing lists and analyze 80 to 100 million pieces of information annually.

SOLUTION: Installed a 45 workstation Novell LAN

BENEFITS: Ability to track responses by a variety of variables.

Five Ways Networks payoff
PC World, March 1991
Sharon Fisher

In a very short time, we have passed from an era where we simply connected PCs in a peer-to-peer arrangement for the purpose of sharing resources like printers. Many companies have taken this a step further, even as far as mission critical applications. John Glaser, Vice President of Information Systems at Brigham and Women's Hospital, in Boston says "If we screw up, it can kill people".

ORGANIZATION: Brigham and Women's Hospital

PROBLEM: Share information across departments
Maintain a very high uptime reliability rate

SOLUTION: Connect data General Minis to a token ring PC LAN, with MUMPS operating system.

BENEFITS: Collect information from a myriad of sections, make one composite picture.

The Lay of the LAN
CIO, September 1991
Leslie Goff

Yes, I do practice what I preach. Shortly after arriving at the Revlon facility in Phoenix, I was faced with a couple of automation issues.

- Information present on multiple computer platforms.
- Growing number of information "Islands".
- How can we standardize on PC software packages and comply with license agreements.
- Provide an alternative to "Sneakerware".

Accomplish the above without installing mini or mainframe hardware was the challenge.

- Applications already existed on DEC PDP's.
- Applications resided on an IBM 3090 Mainframe.
- Stand alone PCs were being utilized.

We had IBM 3274 mainframe terminals, DEC VT100 terminals and PCs and the like on desk tops. Multiple terminals for a single user was common. I had vision and still do. Homogenize this potpourri and create a network environment that will not only solve these problems but provide a foundation for the future.

I will move the clock ahead five years, the current network consists of:

- 250 PC's on a LAN
- 8 file servers bridged into a single network, 8 giga bytes of online "mirrored" storage.
- 3 Gateways to a IBM 3090
- Linkage into a Stratus mini computer
- Access to the DEC PDP platforms
- HP 980/300

We are running applications including simple departmental data bases, bar coding data collection, RF (Radio Frequency) Terminals, and Material Requirements Planning (MRP).

SUMMARY

In the limited time available I have tried to expose you to an alternative automation architecture. A new phrase **SEAMLESS ACCESSIBILITY** was coined. The problems facing information and business management today, as we become aware of the reality that our company's informational assets are residing throughout the enterprise without control was briefly examined. Finally we took a look at some of the technical aspects of **NETWORKED COMPUTING.**

This entire server based computing architecture, historians tell us, I use the word historian loosely, began way back in 1979. In a secret research facility of Xerox, a new way of building a computer was being conceived. The central theme was to slice up the power and function normally encapsulated in a mainframe or mini, spread them out and connected by a network of some type. Have dozens or even hundreds of small computers doing dedicated functions or performing processing tasks on the desk top. Steven Jobs coined a phrase that defines this topic, "THE NETWORK IS THE COMPUTER".

The year of the LAN has occurred for many years running. I do believe that Novell has brought it home to roost in the DOS environment. Charles E. Exley Jr. CEO of NCR Corp states "Ultimately, every business will want to move from their mainframes." "Servers are going to blow the doors off of traditional mainframes says CEO Larry Boucher of Auspex Corporation.

NCR's commitment to the server concept is so strong that they have based their entire future product line on the Intel chip. This strategy caused a vicious hostile take over by AT&T, in order to acquire NCR's position. Can a micro chip handle the volumes of large corporations? NCR is testing computers that could deliver 240,000 MIPS. That is not a typo folks, 240,000 MIPS on a micro processor based computer. HP has done a great job of embracing the OPEN environment.

Predictions of the "HOT" server market which was a 3 billion dollar industry in 1989 is good. This industry is projected to grow to a little short of 12 billion by 1994. This growth is fueled by companies like HP, NCR, Compaq, Sequent, Auspex.

REMEMBER! The **SEAMLESS ACCESSIBILITY** model can be embraced in a incremental fashion. It can leverage your current hardware investment in mini's, mainframes and PC's while solving problems from productivity, capacity and overcoming information "Islands".

Today, as we are leaving behind the age of centralized, mainframe computing and entering an age of distributed computing, intelligence can reside at any point on the network. Many companies are choosing this alternative. Seventy six percent of the I.S. of the 110 companies surveyed by CIO magazine said they have implemented distributed computing or are planning to do so soon.

Putting An HP3000 on the Desktop

Breaking Down the Client/Server Barriers. Integrating MPE/iX, Posix, NetWare/iX, and AppleTalk/iX Files on the HP3000.

Paper # 7023

Bill Lund

Hewlett-Packard Company
Information Networks Division
19420 Homestead Road
Cupertino, CA 95014

(408) 447-2450

1. Abstract

With the MPE/iX 4.7 release of NetWare for MPE/iX (NetWare/iX) and AppleTalk Services for MPE/iX (AppleTalk/iX), MPE and Posix users, have for the first time direct access to data and files created and stored by network operating systems (NOS). NetWare/iX is the MPE/iX implementation of Novell's NetWare for Unix version 3.11a, and AppleTalk/iX is the implementation of Pacer Software's portable AppleShare compatible server. This paper will discuss how to share files between NetWare/iX, AppleTalk/iX, and Posix aware MPE/iX applications, how to span the naming conventions of each NOS, and discuss differences in data format. We will also cover aspects of file security as implemented by AppleTalk/iX and NetWare/iX to manage shared files such that security is maintained across NOS access.

2. Introduction

As the spread of desk top computing continues to pick up speed and importance, including the HP3000 as an integral part of the computing environment becomes critical to making effective use of your data and applications. The HP3000 provides efficient and secure means to store and manage large amounts of data, which, along with industry leading applications makes access the HP3000 from clients an important factor in a complete information systems plan.

Many attempts to jump directly into distributed applications by using client/server APIs such as NetWare's SPX, have been frustrated by the high cost of redesigning and rewriting existing applications to split data display, data manipulation, and data storage to different platforms. Although the future continues to hold the promise of distributed cooperative computing, realization of near term benefits requires the use of data access techniques from the client to the server.

We will discuss how to use the NetWare/iX and AppleTalk/iX products to give effective access from NetWare and Macintosh clients to data stored and created on the HP3000. This is the next step toward desk top integration and cooperative computing.

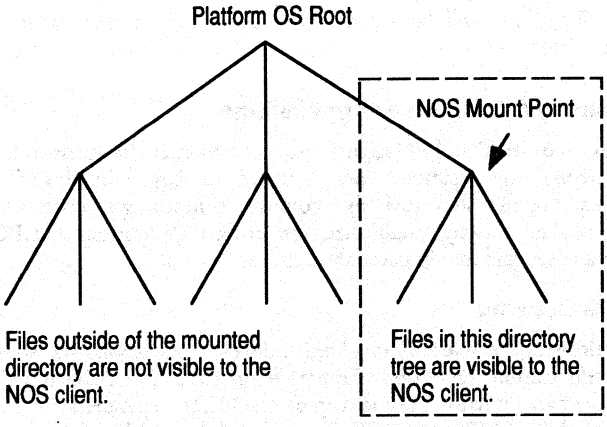
2.1. The Dangers of Early Information

The versions of NetWare/iX and AppleTalk/iX discussed in this paper are scheduled to be shipped with MPE/iX release 4.7 in early 1994. The information presented here is accurate at the time of this writing, but is subject to change as the development process for these products continues. The reader should use this information to conceptually understand and devise how he or she will make use of these capabilities. Specific implementation details should be reviewed at the final release of the products and could be different than what is discussed here.

2.2. Overview of Networked Operating Systems

Networked Operating Systems are additions to the fundamental capabilities of a system. Normally, the NOS is either added on to the operation of an underlying operating system (OS), such as the addition of AppleShare services to the Macintosh OS, or they constitute the entire OS of the platform, such as the NetWare on Intel which replaces DOS entirely. In the case of MPE/iX, NetWare/iX and AppleTalk/iX are both additions to MPE and can run concurrent to each other and other networking products.

When a NOS is an addition to the platform's existing OS, such as NetWare/iX on MPE/iX, the underlying file system is made available by defining a mount point. This is the "root" of the NOS directory and all files and directories of the server which are within the mounted directory are visible to the NOS client. Conversely, all files which are outside, either above the mount point in the directory structure or on a different branch, are not visible to the NOS client. If NetWare/iX and AppleTalk/iX both define the same mount point, then the files stored in the mounted volume will be visible to both NetWare and Macintosh clients, as well as to Posix aware MPE/iX applications.



2.3. Overview of NetWare/iX on MPE/iX Release 4.7

NetWare/iX for MPE/iX release 4.7 is a new ported version of the latest release of NetWare for Unix (NWU) from Novell. Besides including the features of NWU version 3.11a, which includes better printer sharing integration and support for the Burst Mode client, HP's implementation of NWU 3.11a will be (along with AppleTalk/iX) the first networked operating system on MPE to directly support native Posix files. Within the naming constraints which are discussed below, files stored in the NetWare/iX volume will be directly accessible and visible to Posix aware utilities and applications. This means that MPE users will be able to directly open and manipulate files which were stored on the HP3000 by a NetWare client without resorting to copying the data out of a closed volume accessible to NetWare only. Further, a NetWare client can directly access a file created by an MPE application, without requiring the file be copied into a NetWare only volume.

Additionally, the significant performance gains released on the currently shipping version of NetWare/iX for MPE releases 4.0 and 4.5, will be rolled forward into this release. The currently shipping version is based on NWU 3.01b and is similar to all previous NOS implementations on MPE by using a closed file volume to store NOS files.

The application program interfaces (APIs) for creating distributed applications will also be available.

2.4. Overview of AppleTalk/iX

Similar to NetWare/iX, AppleTalk/iX is based on a port of a Unix implementation of an AppleShare server and will use the Posix file system to manage files created by Macintosh users. Since virtually all printers in the Macintosh environment are networked, there is no need for printer sharing services. AppleTalk/iX is based on the latest AppleTalk protocol version from Apple Computer and will ship on MPE/iX

release 4.7. The APIs will be available by special arrangement with Information Networks Division.

3. Building Client/Server Applications

The integration of MPE and PC clients into a common environment can take several forms stretching from simple data copying to large distributed computing environments. The first steps towards a common computing environment start with the sharing of data between established and proven applications and PC programs designed to analyze and manipulate information.

3.1. Data Copying

Until this release of NetWare/iX and AppleTalk/iX, data copying was the only means to retrieve information stored on MPE for use on a PC. The characteristic feature of this is that the data on MPE and the data on the PC are separate, having been copied or down loaded from MPE to the PC. Examples of this would be the use of products like Information Access or a terminal emulator to copy data from a flat file or structured file such as an Image data base to a file on the PC. At the time of copying, the files are synchronized and the user is assured of consistency. As soon as the copy completes (or even before if concurrent access is possible) the files are potentially no longer synchronized.

Data copying is only appropriate when the frequency and amount of data involved is small otherwise the cost of executing and supporting data copying as a integration strategy becomes too high. The full costs need to include an analysis of the dangers inherent in the lack of synchronization as well.

3.2. Data sharing

The real value of NetWare/iX and AppleTalk/iX on MPE 4.7 is their ability to allow the direct sharing of data between the clients and the server. The problem of synchronization is reduced in that the clients do not work on a copy of the data found on the server, but on the server itself. The application designer needs to insure that the data being shared on MPE is found in directories which are mounted by the network operating system, and is not in a structured file, such as a KSAM file, which cannot be easily read by PC applications.

3.3. Client to Server

The advantages to true distributed applications are that only the data required by the PC user is accessed, and the platform best suited to the task is the one performing the task. For example the PC performs local data manipulation, executing the analysis which the user specifically wants. The MPE system manages the large data file and responds to specific requests. The problem is in the cost of developing distributed applications.

4. Data Sharing

4.1. Using the Posix File System as a Networked Operating System Volume

The most important feature of NetWare/iX and AppleTalk/iX on MPE 4.7 is the openness of the file system and the ability to access PC files both from the client and from an MPE/iX application or utility. As could be expected the differences between the user definitions, the character sets and format of file names, and security need to be considered when crossing over between DOS, Apple, and MPE.

4.1.1. Name Space Convergence

4.1.1.1. Definition

One of the characteristics of a file system is the means by which the user identifies files, typically called the "name space". When converging more than one file system as represented by a network operating system, the collision of name spaces has to be resolved to allow files from one space to be visible and accessible to the other. The issues which need to be dealt with include file name length, mapping of characters available in one file system to the characters available in the other, and dealing with the collision which occurs when two files from one name space map to the same name in an other name space.

Additionally, files contain information which deal with file owners, file creators, and format as well as the concept of file forks containing separate file partitions. How this is represented or not represented across NOS domains is important to successful access. In the case of NetWare and AppleTalk on MPE/iX, file forks and labels are stored in specialized files and directories called shadow directories because they shadow the underlying file system directory and add to it.

Typically users defined on one network operating system will not necessarily be known to other NOSes.

4.1.1.2. Name Space Examples

4.1.1.2.1. MPE

The MPE name space for files and directories is fairly simple. The maximum name length is 8 characters. All alphabetic characters are upper case. The initial character must be numeric. The remaining characters are A–Z and 0-9.

MPE Name Space

Character set	`A-Z, 0-9`
Format	8 characters maximum leading alphabetic character examples: `FILENAME` or `A2837234`

4.1.1.2.2. Posix

The Posix name space on MPE/iX release 4.7 is slightly larger than MPE. The legal characters are a-z, A-Z, 0-9, and ".", "_", and "-".

Posix Name Space

Character set	`A-Z, a-z, 0-9` `. _ -`
Format	file name 256 characters maximum path length 1024 characters maximum example: `FiLeNaMe_WhIcH_Is.LONG_LONG_LONG`

4.1.1.2.3. DOS

The DOS name space is well known and consists of A-Z, 0-9, and the following non-alphabetic characters: $, &, #, %, ' (apostrophe), !, (,), -, _, @, ^, {, }, ~, and ' (single quote). It should be noted that lower case letters are allowed in DOS, but are up shifted and then acted on.

DOS Name Space

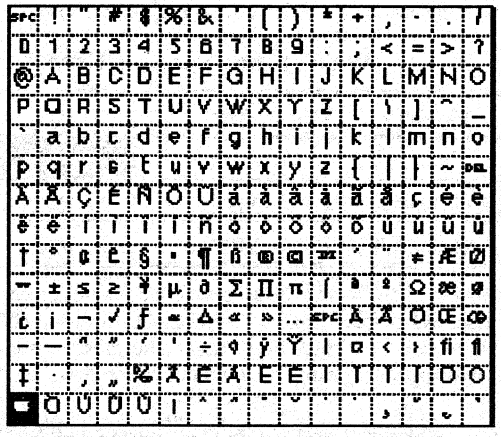
Character set	`A-Z, 0-9`
	`$ & # % ` ' ! ( ) - _ @ ^ { } ~`
	Note that lower case characters are allowed in DOS commands, but are shifted up before being acted on.

Format	file name 8 character maximum with 3 character extension eg. `FILENAME.EXT` or `FILE$$.#01`

4.1.1.2.4. Macintosh

The Macintosh name space is significantly larger than DOS or Posix. The screen capture below shows all Macintosh characters. This is further complicated by the ability of the Macintosh to assign other character sets or fonts to the file naming space. For example, the Japanese version of the Macintosh OS uses Japanese characters for file names. In this case information about the font being used is stored by the server.

Macintosh file names preserve upper and lower case characters but for the purposes of defining a unique name, upper and lower case characters are the same. For example, the file "aaa" and "AAA" could not appear in the same directory since they would have identical names for directory purposes.

Character set	
Format	Macintosh file names have a maximum of 32 characters where upper and lower case characters are preserved but do not constitute unique file names. Example file name: `Bill's Report from München`

4.1.1.3. What is a "well behaved" file name?

When crossing over from one NOS name space to another, the differences between the two spaces must be resolved in order to display the source file names in the destination name space. For example, Posix file names can be much longer than the eight character limit imposed by DOS. When a NetWare client is attempting to enumerate or access Posix files, some type of mapping must occur in order to display the Posix files on the NetWare client which do not conform to the DOS name space. The details of the mapping will be discussed below, but for now let's discuss how to avoid mappings altogether. This is called a "well behaved file name."

In general a well behaved file name is one which requires no mapping across NOS domains. In other words it conforms to the most restrictive name space and mapping is not needed. From the tables above we see that DOS uses A-Z, 0-9 and some special characters, and that the format of the file name is up to an eight character file name, optionally followed by a period and up to a three character extension. This format is more restrictive than the Posix format, or in other words, all DOS file names are legal Posix file names. The converse however, is not true. Not all Posix file names are legal DOS names. So if we are concerned about accessing Posix files from a NetWare client, we should limit the files on the Posix side to names which conform to the DOS name space. This way no mapping from Posix to DOS is needed.

The matrix below shows which name space is most restrictive. For example when mapping from Posix to MPE, MPE is the most restrictive and if the MPE file naming conventions are followed, no mapping occurs.

Which Name Space is More Restrictive

From/To	MPE	Posix	DOS (NetWare)
Posix	MPE		
DOS (NetWare)	MPE	DOS	
Macintosh	MPE	Posix	DOS

In general, MPE is the most restrictive name space with DOS the next most restrictive. When planning to create applications which will cross name spaces, it is valuable to use files which conform to the most restrictive space. Well behaved file names will appear the same in any name space view.

4.1.1.4. What is the mapping for "ill-behaved" file names?

The Posix file system is the common factor in the NetWare, AppleTalk, and MPE files. Both NOSes store all files in the Posix domain and consequently have to map their name spaces into the Posix name space. For this reason we will only consider the mappings to and from Posix. Since at this release there is the convergence of NetWare and AppleTalk through the Posix file system, a NetWare file seen from AppleTalk will appear as a standard non-AppleTalk Posix file. Likewise, an AppleTalk file will appear to the NetWare server as a standard non-NetWare Posix file and will be treated as such. A discussion of file forks and shadow directories will be found later.

4.1.1.4.1. MPE to Posix

The mapping from MPE to Posix is fairly simple. Since MPE file names are a subset of the full Posix name space, MPE file names are preserved if copied into a Posix directory. Likewise, MPE file formats are preserved.

4.1.1.4.2. Posix to MPE

The only Posix to MPE mapping that occurs is when an MPE application accesses a Posix file. Posix files can exist either within a Posix directory, or within an MPE group and account. At this writing no mapping occurs in that an MPE application or command is either Posix aware or it isn't . If it is Posix aware, then the Posix file and directory names will be visible without mapping. If it is not Posix aware, then the Posix files and directories will be invisible.

4.1.1.4.3. NetWare (DOS) to Posix mapping

When NetWare/iX stores a NetWare file on the MPE server, the file name must be mapped to a legal Posix file name. The NetWare/iX server will preserve the DOS or NetWare name of the file, and store the file in a legal Posix file. When mapping from DOS to Posix the server needs to map the special characters of DOS to a legal Posix

file name. As a developer you need to know the mapping so that you can identify the Posix file which maps to the NetWare (DOS) file you are after.

The mapping is displayed in the table below. Each DOS special character is mapped to a three character sequence. One implication of this is that the length of the name from the Posix perspective is longer than the file name from the DOS perspective.

DOS to Posix Character Mapping for File Names

DOS	Posix
!	..1
@	..2
#	..3
$	..4
%	..5
^	..6
&	..7
(	..9
)	..0
`	..a
'	..b
{	..c
}	..d
~	..e

For example the DOS file called "`FILE$$.#01`" would be mapped to the file called "`FILE..4..4...301`".

From the NetWare client, a file in the NetWare domain appears with its NetWare (or DOS) name. From a Posix aware application, or from the AppleTalk/iX server, the file name would have the mapped name as described above.

4.1.1.4.4. Posix to NetWare (DOS)

A Posix to NetWare or DOS mapping would occur when a Posix aware application or another NOS placed a file into the NetWare mounted volume. In this case, the file would appear to the NetWare/iX server. The Posix to NetWare mapping is much simpler than the DOS to Posix mapping. Since Posix files can contain both upper and lower case letters, the only mapping is that lower case letters are mapped to upper case. For example the Posix file called `file.txt` will appear as `FILE.TXT`.

There is a case where two Posix files will map to the same NetWare file name. For example, `file.txt` and `File.txt` will both map to `FILE.TXT`. The file with the upper case will take preference and the other file will be unavailable from the NetWare side. In this example, `File.txt` will map to `FILE.TXT` and `file.txt` will be inaccessible from the NetWare client.

4.1.1.4.5. AppleTalk to Posix mapping

The Apple Macintosh file name space is considerably larger and more diverse than the DOS name space. Macintosh files are limited to 32 characters, but the character set is quite large. Below is a partial mapping of Macintosh characters to Posix characters.

Macintosh to Posix Character Mapping for File Names

Macintosh	Posix
ä	a
ö	o
ü	u
ß	B
Ä	A
Ö	O
Ü	U
fi *(ligature)*	fi
fl *(ligature)*	fl
space	_ *(underscore)*
Ω	o

Because the AppleTalk/iX server maps more than one Macintosh letter into the same Posix letter, there is the possibility of collision. For example, the Macintosh file names "aBc" and "äßc" would both map to the Posix file name "aBc". When this occurs, one of the file names has a number appended to it. In our example the file "aBc" could be mapped to "aBc" and the file name "äßc" would be mapped to "aBc1". From the Macintosh client, both files would retain their Macintosh names.

4.1.1.4.6. Posix to AppleTalk/iX

The two cases where Posix file names are mapped to AppleTalk/iX names is when an application puts a file into the mounted AppleTalk/iX directory and when NetWare and AppleTalk share a common mount point. In the first case, since Posix file names are legal Macintosh names no change is made. In the second case, the NetWare to Posix mapping may have resulted in a different name. This Posix mapped name is the one which will be displayed to the Macintosh client. For example, a DOS file named `FILE$.TXT` would be mapped to `FILE..4.TXT` by the NetWare server. If this file were visible to the AppleTalk/iX server, either by being copied into the AppleTalk mounted volume or by AppleTalk and NetWare sharing a volume, then the file would appear to the Macintosh by its Posix name, rather than its NetWare name.

Another factor in mapping Posix to AppleTalk is that Posix file names can be as long as 256 characters. The Macintosh only supports file names of 32 characters. The

AppleTalk/iX server will truncate the Posix file name to 32 characters. In the case where two Posix files will now have identical names in the Macintosh name space, only one of the files will be visible and accessible.

4.1.1.5. File Forks and Shadow Directories

Both NetWare/iX and AppleTalk/iX need to support file forks and file attributes which are not native to MPE/iX or Posix. The servers do this by providing shadow directories and files which under normal circumstances are not visible to the client. A shadow directory is a file containing directory, attribute, or file fork information.

4.1.1.5.1. NetWare

NetWare/iX version 3.11 maintains an "`inode`" file in each directory under the mounted directory of the NetWare volume with information about every file in the directory. The `inode` file contains NetWare attributes (eg. `SYSTEM`, `COPY INHIBIT`, `SHAREABLE`), mapping between NetWare/DOS file name and Posix file name (eg. `FILE$.TXT` maps to `FILE..4.TXT`), and NetWare trustee information for files and directories.

These `inode` files are not visible to the NetWare client, however, since they are Posix files, they are visible to any Posix application or command, and to AppleTalk/iX if it shares a mount point with NetWare/iX. Care should be exercised not to alter any of these files since corruption of the NetWare server volume can result. The `inode` files are owned by the NetWare/iX server and are not accessible by any other user with normal capabilities.

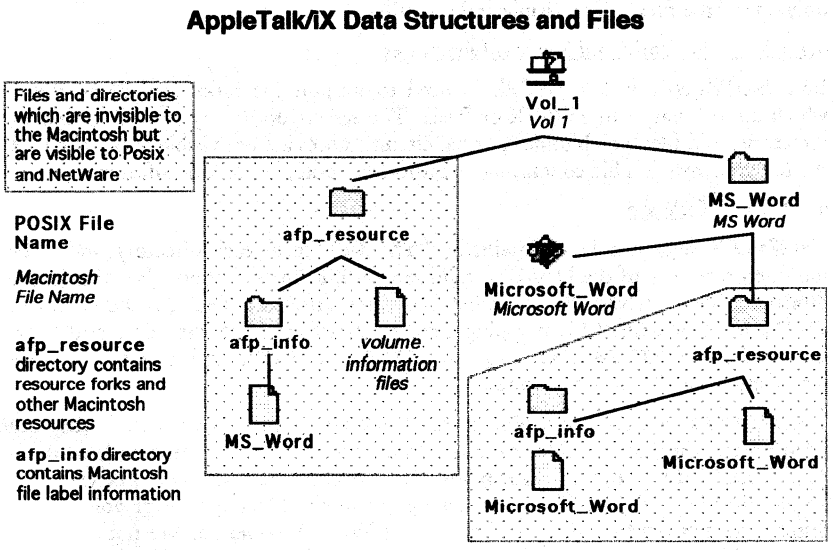
When a Posix application or the AppleTalk/iX server places a file in the NetWare/iX volume, the file will at first not appear to the client. The server will scan the current active directory of the attached NetWare client, and when a file is found which does not have an `inode` entry, one will be created for it with default attributes and trustees of the directory it is found in.

4.1.1.5.2. AppleTalk/iX

AppleTalk/iX has a more complex structure. In addition to supporting attributes not found in Posix, the Macintosh supports files with data and resource forks, as well as information on the type of file, the application that created it, and the icon associated with the file.

The drawing below illustrates the shadow directories and files found in the AppleTalk/iX server. Each visible directory has a sub-directory called "`afp_resource`" which contains the resource fork for the files in that directory. For example, the directory "`MS Word`" contains a file "`Microsoft Word`" and the `afp_resource` shadow directory. In the `afp_resource` directory the resource fork for the file "`Microsoft Word`" is found, and has the same name as the visible file in the directory "`MS Word`". The data fork for "`Microsoft Word`" is the visible file in the "`MS Word`" directory. Within `afp_resource`, there is another shadow directory called "`aft_info`" in which the Macintosh attributes and security information is kept. The icons for the volume are found in the

"afp_resource" shadow directory in the root of the mounted volume, in this case the mounted volume is called "Vol 1".



In this example, the only files and directories visible are the volume "Vol 1", the directory or folder "MS Word", and the file "Microsoft Word". All other directories and files are not visible to the Macintosh client. However, as with NetWare, since the shadow directories and files are Posix files, they are visible to Posix and NetWare applications. Care should be exercised to not modify the shadow directories and files since corruption of the server could result. The shadow directories and files are owned by the AppleTalk/iX server and are not accessible by any other user with normal capabilities.

4.1.2. Data Format of "Flat" Files

Since sharing files is one of the advantages of new releases of NetWare, AppleTalk, and Posix on MPE/iX release 4.7, the actual data format of files is of interest. NetWare and DOS, MPE, Posix, and the Macintosh file systems each represent flat files in different formats. A flat file is one in which no structure, such as pointers or indices, is present.

4.1.2.1. Differences between file systems

The Posix byte stream file, which is equivalent in function to unstructured DOS or Macintosh text files, is a continuous stream of bytes. The separation between records is indicated by the <line feed> character. There is no mechanism to uniformly identify where in the file a given record will occur. Similarly, the Macintosh file system indicates the end of a record in a text or flat file with a <carriage return> character.

DOS and NetWare differ from Posix and the Macintosh file systems in that two characters, <carriage return><line feed>, are used as a record terminator.

When sharing files across network operating systems, you will need to resolve the differences between the expected end of line characters. Most current applications, such as Excel or Microsoft Word are capable of understanding these differences when reading in a file created on another operating system. However, if you are building your own applications which intend to share files across platforms, you will need to account for this difference. For example, a byte stream file created by a Posix aware application on MPE/iX will use <line feed> to delimit records. If you intend to read this file from a NetWare client, the application reading the file should expect to see <line feed> instead of <carriage return><line feed> as the delimiter.

4.1.2.2. DOS to non-DOS character mapping problem

Another problem which will be encountered when moving between DOS and non-DOS network operating systems is the expected location of records within the file. Since DOS and NetWare use two characters to delimit a record, and Posix and Macintosh use one, any program which attempts to locate a specific record in the byte stream by calculating the byte position (assuming that each record has the same number of bytes) will need to accommodate the different number of record delimiter characters.

4.1.3. Access to "Structured" Files

At this writing, AppleTalk/iX will be able to access MPE files which are either fixed or variable record format. The access to these files will emulate access to Posix byte stream files, appending the Macintosh record terminator character to designate record boundaries. The files themselves will not be converted. MPE structured files, such as Image or KSAM files will not be accessible from AppleTalk/iX. NetWare/iX will only directly access Posix byte stream files. Allbase files will be accessible programmatically over the NetWare transport and APIs via the Client/Server Allbase product.

4.2. Administration and Security

4.2.1. NetWare Users

NetWare/iX supports the standard mechanisms for defining NetWare users and trustee rights to files and directories on the server. From the NetWare perspective this is identical to what would be expected of an Intel version 3.11 server. From the MPE side, all NetWare/iX files, belong to one user, NWUSER.SYS. One of the configuration parameters is an "open mask bits" which is used to define the default MPE security of all NetWare/iX files. This can be defined such that only NetWare/iX can have access, or that any SYS account user can have access, or that the files are accessible from any user. Once a file is created, the Posix utility CHMOD can be used to individually change the security of a file. Care should be taken to ensure that the NetWare/iX server still has access to the files.

4.2.2. AppleTalk Users

AppleTalk/iX uses the MPE system security and users to define access to the server files. Any MPE user may access the AppleTalk/iX server by providing the appropriate passwords. The files which are stored on the server from the Macintosh client are associated with the MPE user which logged on to the server. An example of this would be an MPE user called `bill.network`. This user would be logged onto the AppleTalk/iX server with access to any file and directory to which `bill.network` would normally have access as an MPE user.

The AppleTalk/iX server also allows a Macintosh user id to be mapped to an MPE user. For example, the Macintosh user "`bill lund`" could be mapped to `bill.network` and have access to all that MPE user's files and directories .

4.2.3. Mount Points

NetWare/iX allows any Posix directory to be used as the mount point for the volume. All enclosed files and directories are then visible to the NetWare client, in so far as NetWare defined security allows access. When the server is first installed a default mount point is defined as `/usr/netware/sys` for the `SYS:` volume. Other volumes may be defined in addition to the `SYS:` volume..

Likewise, AppleTalk/iX allows any Posix or MPE directory to be used as a mount point for the server volume. During installation the administrator has the option to define a mount point, but if none is chosen, a default mount point of `/users` is defined as the `Users` volume. Other volumes may be defined.

4.2.4. File Locking

AppleTalk/iX will provide a means to configure concurrent locking of AppleTalk and Posix files. This is known as "bleed through locking." When configured, the server will use an MPE lock when the Macintosh client locks a file on the AppleTalk/iX server. Likewise, when an MPE application locks a file visible to the AppleTalk/iX server, it will be locked to the Macintosh client. NetWare/iX will support this in a future release.

4.3. Convergence. Mounting the same Posix directory space by NetWare/iX and AppleTalk/iX

What should you think about if you intend to create a shared volume? We've covered file naming, data formats, and security. A successful plan for providing true cross platform access needs to consider first the NOSes which will be supported then look at the requirements for sharing data. The case of sharing Posix and one NOS is fairly well described above, let's consider the case of sharing a single mount point for NetWare/iX and AppleTalk/iX so that Posix, NetWare and AppleTalk have common access.

4.3.1. File Name Mapping

The easiest way to insure that file names are reasonable and understood is to use the most restrictive naming space, which in this case is DOS. If all shared files and directories conform to the DOS name space they will be well behaved across all three name spaces.

4.3.2. Data Format

The common format for all three platforms is a byte stream file, taking into consideration the differences in record terminators.

4.3.3. User Names, File Owners, and File Access

This is the most complex issue, since AppleTalk/iX and NetWare/iX have very different architectures for providing security. Clearly, the first step in providing access across all three environments is to create a common mount point by defining the same Posix directory as both a NetWare/iX and AppleTalk/iX volume. Although the shadow files and directories of each NOS will be visible to the other, they will be inaccessible.

How do we provide file access? There are two ways, the first is to make an AppleTalk/iX user which is mapped to `NWUSER.SYS`. Since this is the MPE user which is identified as the owner of all NetWare/iX files, all files in the shared mounted volume will be accessible to a Macintosh client. From the NetWare client side, the trustee assignments for the volume will be in place so that only the files which NetWare/iX grants access to will be available to NetWare. Access from the Macintosh client should be controlled by only placing those files in the shared volume which should be shared by the Macintosh. Since only those directories which are explicitly identified as AppleTalk/iX volumes are mountable from the Macintosh client, the rest of the NetWare/iX volumes are safe from access by the AppleTalk/iX server.

The second way to provide controlled access would be to create a user called `APPLE.SYS` and then explicitly share NetWare/iX files and directories with this user. The Posix utility `CHMOD` provides a way to change the access rights on a Posix file such that other members of the Posix group have access to the file. All users within an MPE account are considered members of the same Posix user group, hence `NWUSER.SYS` and `APPLE.SYS` would be members of the same Posix group. By using `CHMOD` to grant access to group members, individual files and directories would be shared between the NOSes. Again, only those directories which are defined as AppleTalk/iX volumes will be accessible, so the rest of the NetWare/iX server would be safe from access by Macintosh clients.

5. Promise of Client/Server

One of the values of the MPE/iX 4.7 release of NetWare/iX and AppleTalk/iX will be to open up the Posix file system to MPE, NetWare, and Macintosh operating systems. The first step in providing true distributed processing and client/server computing will be to share the data between platforms, bringing the data and applications of the HP3000 to the user's desk top.

Conquest

An Object-Oriented

HP Configuration and Quote System

Bob Lewis and Susan Stavish

Hewlett-Packard Company
19490 Homestead Road
Cupertino, California 95014

Abstract

Hewlett-Packard encourages creativity in its product divisions while also providing customers with integrated system solutions. The resulting diversity among the divisions, however, loads Hewlett-Packard sales people with confusing configuration rules.

Currently, Hewlett-Packard prepares system solution configurations and accurate price quotations only with great difficulty. This process often takes several days involving many people to catch and correct errors because of the complexity of the product line.

To solve these problems Hewlett-Packard developed Conquest, which stands for CONfiguration and QUotation Sales Tool.

Conquest is a client-server application that uses a structured database to store the configuration rules for all Hewlett-Packard products. It permits Hewlett-Packard sales people to prepare their own complete and accurate quotations. Conquest also links to the legacy systems essential to smooth manufacturing and order processing in order to retrieve current prices.

This paper describes the history and development of Conquest. We discuss how Conquest works and how the knowledge database is maintained.

Introduction

Hewlett-Packard's customers have consistently rated order fulfillment as their number one area of dissatisfaction. And customers' expectations are increasing — they want an even broader and more rapid availability of products. Hewlett-Packard sales representatives are also unhappy with the order fulfillment process. They say that they spend from 30% to 40% of their time on the configuration and quotation process, which is the important first step in providing superior business solutions. Studies suggest that rework caused by errors and delays at this stage incur costs in the millions of dollars each year.

This paper describes the Hewlett-Packard response to these deficiencies. We first look at the organizational response and the re-engineering of the current configuration and quotation process. We review CONRAD, the CONfiguration Advisor tool. Then we examine the latest contribution to the solution — a client-server, expert system tool called Conquest (CONfiguration and QUotation Sales Tool).

Organizational Response and Process Re-engineering

> "By the end of 1995, Hewlett-Packard needs to achieve a company wide tenfold improvement in the order fulfillment process as measured by our customers."

This statement is one of the two key goals of Hewlett-Packard CEO Lewis Platt, and thus represents the highest level of commitment to solve the order fulfillment problem. To ensure that we achieve this goal, a Vice President of Order Fulfillment was recently appointed, and a Configuration/Quote Task Force was formed to improve the configuration and quotation process. The task force's goals are:

- Reduce the time and work sales representatives spend on the configuration and quotation process.
- Reduce the time and work Hewlett-Packard people overall spend on the configuration and quotation process.
- Increase the percentage of times Hewlett-Packard meets deadlines for customer quotations.
- Improve the accuracy and presentation of price quotations.

In short, the goal of this task force is **to make it easier for Hewlett-Packard customers to do business with Hewlett-Packard.**

Early Process Improvements

Observers often ask why Hewlett-Packard's product structure and price list grew so complex. Simply, the product structure is complex because Hewlett-Packard manufactures a large number of products and peripherals, and offers the customer numerous configurations to choose from. To provide each customer with accurate product quotations, and valid orders, Hewlett-Packard established a long list of rules to guide sales representatives and others involved in the process.

The product structure schemes were developed and are updated in part to recognize the needs of the manufacturing divisions which must solve problems involving inventory control and parts allocation.

One recent change in the Computer Systems Organization product structure that was made to facilitate manufacturing involved replacing the numeric model numbers, such as HP9000 Model 827 with names such as HP9000 Series G Class Business Server Model G30. Also, system components such as memory boards and chassis were given separate product numbers and now must be explicitly ordered with each system.

Under the old numeric scheme, identical components allocated to different products often could not be interchanged. A memory board in the given system would have a part number unique to that system even if physically identical as memory boards used in other systems. Manufacturing could not borrow parts from one system on the line to help complete another. The new part numbering system permits exchanging parts between systems.

Hewlett-Packard uses system bundles as another simplification approach. These bundles incorporate our most popular components and options under a single product number with an attractive price. This idea fell short because many of the customers who ordered bundled systems often wanted to change one or two minor components. The extra effort needed to do so eliminated the cost savings of the bundling concept.

The legacy systems used to enter quotations into the manufacturing order process must continue to function as they exist today. Certainly, system enhancements were introduced from time to time. But replacing these systems would disrupt existing processes and require retraining Hewlett-Packard order processing and manufacturing people around the world. To meet these needs without completely replacing the legacy systems, Hewlett-Packard developed two new applications — CONRAD and Conquest.

Hewlett-Packard prides itself on providing superior products and solutions to customers. We consider offering flexible system configurations to meet the varying needs of problems and opportunities essential to success.

CONRAD — First Tool

To provide both flexibility and cost savings, Hewlett-Packard developed CONRAD, an AI expert system that applies a series of rules against a list of product numbers and advises the submitter of errors and possible oversights. The CONRAD project included building a support organization that added all product introductions and product structure changes into the CONRAD rules as soon as they became effective.

CONRAD steps the user through a series of menus, where he selects products and appropriate options for each component of the system being ordered. The list of items is then sent to a checker sub-system which returns a list of suggested changes and error messages.

CONRAD succeeds in helping to ensure that only correct orders reach the factory. After introducing CONRAD, the number of systems that arrived at customer sites with problems such as incorrect cables or peripherals dropped substantially. In fact, some Hewlett-Packard sales regions now require that CONRAD be used to check all quotations and incoming orders.

Despite the value of CONRAD, the tool has several limitations. First of all, the user must understand the product structure and the numbering system as well as configuration rules. CONRAD does not make selections for the user, nor does it free the user from knowing the complex product structure. Secondly, CONRAD does nothing to improve the quotation system. It basically compiles user product number selections and reports conflicts based on a library of rules. Only skilled quotation developers can use this system, and a different system must be used to produce the final quotation.

Thus, Hewlett-Packard decided to launch a multimillion dollar project to replace CONRAD with a more sophistocated configuration/quotation tool called Conquest.

Conquest — The Next Generation Tool

Conquest is a new configuration and quotation system designed to be both powerful and easy to use. Its development has been driven by management's recognition of the issues facing our worldwide sales force and business partners. Coupled with the re-engineering of the core HP configuration and quotation process, Conquest will provide HP with tools to compete in current and future markets.

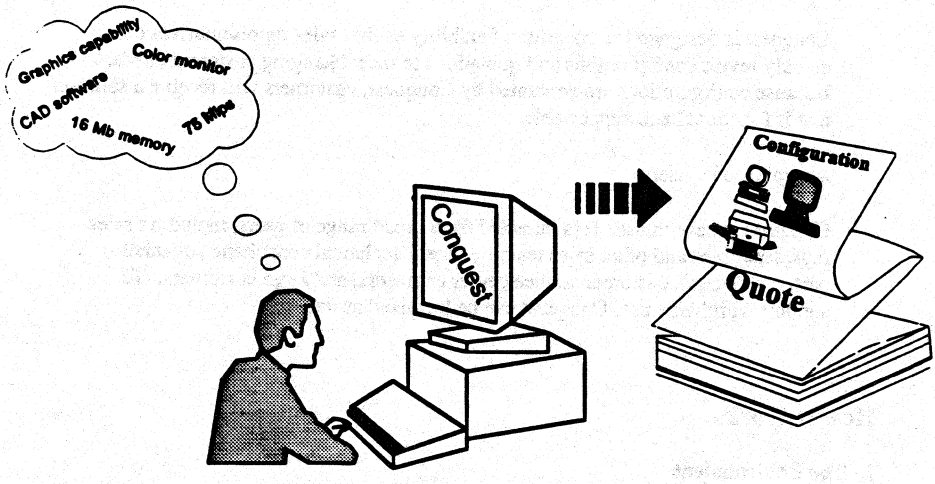


Figure 1. — Conquest

Conquest is designed to meet five general objectives

- Higher sales team productivity

  Conquest significantly reduces the time necessary to produce configurations and quotations, enabling sales representatives to spend less time in the office and more time calling on customers.

- Improved accuracy of configurations and quotations

  Conquest is continually updated with the latest configuration algorithms, prices, and discounts.

- Reduced costs

  Conquest ensures that configurations are complete and functional so that sales team members spend less time reworking solutions and quotations. Because Conquest is linked to order management systems, configurations and quotations will be efficiently integrated into contracts and sales orders.

- Improved customer satisfaction

  Conquest is designed for maximum flexibility so that sales representatives can quickly revise configurations and quotations to meet changing customer needs. Because configurations are generated by Conquest, customers will receive a solution that is functional and supportable.

- Acceptance by users

  Conquest is easy to use. It is intended for a broad range of users, including sales representatives and other sales team members, technical consultants, channel partners (VABs), customer service center engineers, and large customers. To support worldwide use, Conquest can be localized as needed.

How It Works

1. The Environment

The Conquest system architecture consists of client workstations and PCs that access servers for processing, configuration information, and pricing data. One Conquest server is used for processing and for the knowledge base. The knowledge base contains configuration information, models, and constraints that are regularly updated. A second server contains price and discount information. This price server is linked to existing HP pricing systems, which continually update prices and discounts. See Figure 2.

At Hewlett-Packard the interface selection process executes on either a PC or a workstation. These local systems receive current interface and product file data through nightly downloading. The configuration process executes on a local or regional server to provide quicker response time. This configuration process is compute intensive and needs a server to provide adequate response.

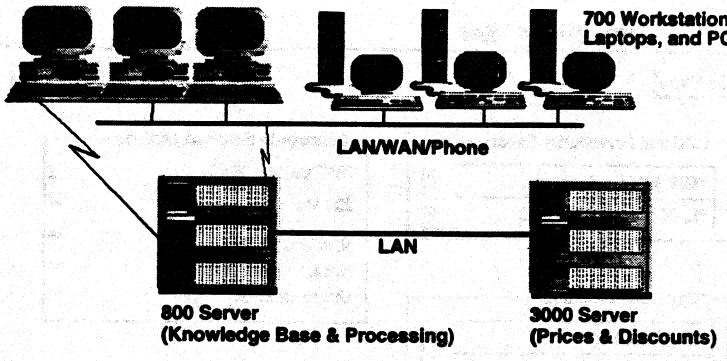


Figure 2. Conquest Architecture

2. Selecting the System

Because Conquest is intended for a wide audience, it has a Windows-based user interface that provides easy access to all capabilities. Conquest users create configurations by choosing from menus and lists. Hardware, software, and customer support options can be selected without ever entering a product number. Since the arrangement and content of menus are customized for each class of system, users select only from options that are valid for the system they are configuring.

For example, the user starts by selecting a platform such as Series 700, 800, or 900. A new window opens, offering a selection of system numbers, such as 827 or G Series, within that platform. The user then selects a system number. First, the introduction screen is displayed, and then the default configuration is displayed. The user can select alternatives to the standard configuration by choosing different processor types, main memory options, console screen colors, country localizations, and so on. If desired, the user can then continue by selecting More Choices.

The More Choices screen shows a list of buttons such as Support, Storage Devices, Networking, and so on. The user traverses a tree of successively narrowing choices until the desired device appears on screen. The device or product appears as a textual description, not a product number. The user selects the desired components by clicking on the list item or entering a number in the quantity box for the desired item. Selecting a group item such as Networking, for example, opens a window showing the categories of networking products, such as Links Without Services, Services Only, and so on. Windows may be revisited to modify earlier choices or to make additional selections. See Figure 3.

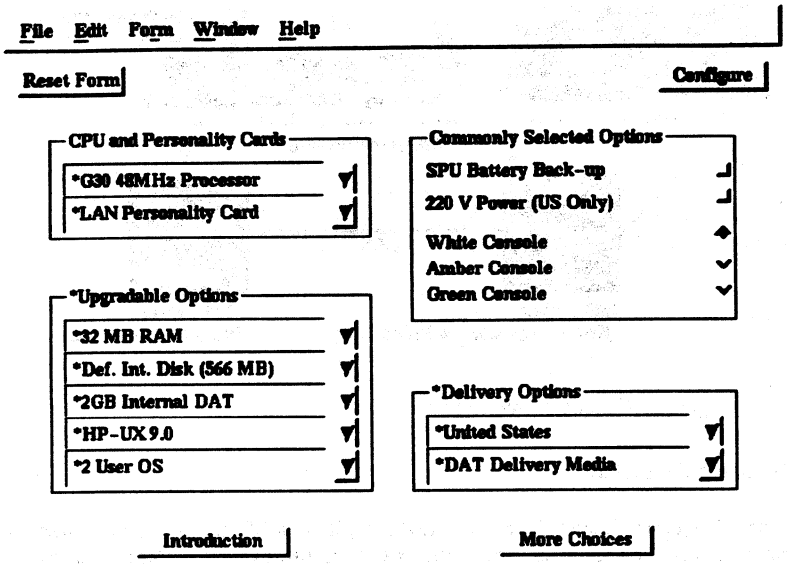


Figure 3. A Conquest Menu

Conquest menus are arranged to allow users quick access to all of a systems configurable options.

The knowledge engineers control the appearance and logical sequence of the selection windows. They can specify constructs for objects including forms, radio buttons, list buttons, list boxes, tables and so on. Simple logic statements control progress from object to object as controlled by user selections and internal controls. When the user selects a given item the component name corresponding to that item is written into the resource request list.

3. Configuring The System

The process just described builds a resource request list. When the system is ready to configure, the user selects Configure System.

At this point the configuration engine takes over and accesses the knowledge base. It uses recursive search techniques to select the series of system components, cables, and

product options that best match the resource request list. Each component selected exists in the database and may include resource requirements that must be satisfied, subcomponents that must also be included, or values that govern subsequent configuration processing. The engine then searches for components that satisfy the requirements of the initial components. As it proceeds, the configuration engine displays its progress and diagnostic messages in the windows.

On completing the configuration, the engine draws a system diagram showing the cabinets, slots, cards, and other components as specified by the knowledge engineers. Peripheral cabinets, modem panels, and so on are displayed as well. Such details as cables are omitted. The user verifies via this diagram that the configuration resulting from the original requests appears to meet the solution requirements.

The user can continue to modify the configuration with the menus even after Conquest has generated the system diagram. He can also make changes interactively by pointing and clicking in the system diagram. When the process is finished the user may request a Bill of Material to check the results of the configuration engine.

The structure and operation of this modeling language is based on the concepts of object oriented programming. Knowledge engineers specify components and build up the system model. Each component belongs to a class which determines properties of the components within that class. Component objects inherit properties from the class of which they are a member. These properties can include such items as resources offered, resources required, attribute values (integer or string) and sub components. The knowledge engineers include components which may not be part of the final quotation, but need to be included in the configuration in order to control subsequent component selection. An example of such a component is a slot into which a given card may be inserted.

Knowledge engineers write statements which specify how to constrain or control the selection of components. These statements look at component properties as described above and govern the selection of components that fulfill requirements such as "requires slot" or "requires connection." Thus the product configuration rules and limitations are enforced.

When new products are introduced the model files generally can be updated by simple cut and paste operations.

4. Generating the Quotation

Once satisfied with the configuration, the user selects Quote System whereupon the engine accesses an HP3000 server via Remote Procedure Calls (RPC) to produce a quotation. This invokes a bundling and quoting process that assembles the specific products and options.

The objective of the bundling is to find a set of products and options that includes all of the required components at the lowest cost. The bundling process uses a segment of the knowledge base known as the Product Component Mapping (PCM) Table. The bundling logic winds its way through the complicated product number and option lists to select those line items required to produce a valid list of products for the quotation.

The PCM files include entries for all available Hewlett-Packard products along with all the available options for each product. The selection of options may be governed by selection of specific components during the resource selection process or may be governed by reference to global conditions such as software media or operating system release level. Other statements control option selection so that,for example, only one of a group such as software media gets included in the quotation.

The quotation reflects the latest HP prices, purchase-agreement discounts, and local options which Conquest retrieves through interfaces to worldwide pricing systems.

Users can choose the level of detail in the quotation, add lines for third-party products, define subtotals, and save formats to use with other quotations. Users can also use the spreadsheet to quote nonconfigured products by entering product numbers and having Conquest retrieve prices and descriptions. See Figure 4.

HEWLETT PACKARD

QUOTE ID: 155
CUSTOMER:
DATE:

1	F_Class1.sb				
1	HP9000/800 F Class Business Server				
1	Model F10 for F Class Business Server	8700.00	8700.00	0.0	8700.00
1	License/Next Day System Support-1st Yr.	474.00	474.00	0.0	474.00
1	MiniNova chassis installed in SPU				
1	Next Day System Support - 1st Yr.				
1	Select ECC SIMM Memory installed in SPU				
1	Standard 16 MB memory module				
1	Next Day System Support - 1st Yr.				

1	Systems Admin Manuals	500.00	500.00	0.0	500.00
1	Advanced Usage Manuals	163.00	163.00	0.0	163.00
1	License to Use System Support - 1st Yr.	105.00	105.00	0.0	105.00

Grand Total: 10604.00

Figure 4. Conquest Quotation

Users can modify how information is organized and displayed in the Conquest quotation.

When the quote has been approved by the customer, it can be exported to the Hewlett-Packard order fulfillment systems that initiates the order filling and manufacturing process.

Maintaining the System — Knowledge Engineering

1. Scope of the Task

Conquest is a great step forward, but it has little value if the configuration and pricing information in the knowledge base is not current. Keeping this information current is the job of the knowledge engineers. Conquest succeeds or fails based on the quality of the knowledge used to maintain the knowledge base. Just as any software system relies on accurate user requirements to succeed, an expert system relies on accurate knowledge. The CONRAD project and other related configuration systems projects pass to Conquest a heritage of knowledge engineering processes and techniques.

Hewlett-Packard management emphasizes the quality of knowledge engineering dedicated to Conquest.

A corps of specialized knowledge engineers work to codify timely and accurate configuration information. To ensure error-free configurations and quotations the knowledge engineers must acquire complete, factual, and accurate information from the product and system divisions.

Knowledge categories:

- Product structure knowledge
- Configuration knowledge

2. Product Structure and Configuration Knowledge

Product structure knowledge is information on what to order to deliver a functional solution including product numbering, required options, optional and delete options, product support service levels, required related products, and optional related products.

Configuration knowledge, on the other hand, is used to assemble and connect systems that work. This kind of knowledge includes the number of peripherals supported, maximum I/O cards, power budgeting and cooling, products and software supported by operating system releases, and memory limits.

Often these two categories overlap; nevertheless they help engineers organize their work. The knowledge base structure is composed of the Product Component Mapping (PCM) segment, which incorporates product knowledge, and the System Model segment, which incorporates configuration knowledge. The PCM data, though generally larger and seemingly more complex, can be derived largely from the existing Corporate Price List systems. These systems are updated and maintained often because the manufacturing process relies heavily on the integrity of this database.

Maintaining the configuration database, however, is more complicated. One reason is that configuration information often changes between releases as a result of test qualification successes and products introduced by peripheral divisions. Gathering accurate and timely configuration knowledge often takes considerable effort.

3. Where Knowledge Engineers Get Their Information

To maintain an accurate knowledge base, knowledge engineers must be aware of both new releases and introductions as well as future product plans and progress.

Knowledge engineers develop most product information through regular contacts with division product marketing engineers who usually work closely with counterparts in the manufacturing divisions while developing the product structure and configuration rules. In some instances, however, the knowledge engineer must interact directly with a manufacturing contact to resolve specific issues.

A number of published information sources also help knowledge engineers gather product information. Often knowledge engineers may obtain early issues of technical datasheets, support publications such as the internal *Hewlett-Packard Computer News* and *Computer Support News*, and bulletins published by the Hewlett-Packard Software Replication and Distribution Operation. The Configuration and Price Guides play a crucial role in maintaining the Conquest knowledge base. Knowledge engineers receive draft copies of these and other product structure and pricing documents as a routine part of the document production cycle.

With the implementation of CONRAD maintenance requirements in 1991, Hewlett-Packard developed an additional check point in the Corporate Price List information flow. Additions and deletions to the Corporate Price List now must be accepted by the knowledge engineers before proceeding through to the Price List Update process. Thus, knowledge engineers can insist on getting adequate information regarding product structure changes before such changes become effective. This step, though bureaucratic, emphasizes the importance placed on the role of knowledge engineering.

Conclusion

Hewlett-Packard has just begun transforming its reputation for inadequate quotation production and order fulfillment. We are well on our way to improving the process by producing quotations that meet the particular needs of each customer while providing the capability to create specific product configurations that are so important to the Hewlett-Packard business style.

Already, Conquest is proving a valuable addition to the Hewlett-Packard order entry process by accelerating the quotation process. It has done so by developing quotations

that are complete and accurate the first time and by developing a focus within the system and product divisions on creating accurate data on the product structure. A substantial savings over the current cost of doing business will result once the benefits of using Conquest become apparent throughout Hewlett-Packard.

Conquest represents a major step. Yet more work remains. The software needs tuning for better performance and adding new features to facilitate ease of use. The Hewlett-Packard worldwide networking and server strategy needs continual review and development to meet the service level needs of our growing business. The knowledge engineering process needs further refinement including involving the system division themselves in database maintenance.

The Conquest user still needs to know how much disk space to order, how much memory to order, and so forth. Potential enhancements include a user profile interface whereupon the user responds to such queries as number of users and size of the database, and Conquest selects system components that best solve the problem.

Hewlett-Packard clearly intends to emphasize Conquest as a major building block to achieving the key goal of making Hewlett-Packard easier to do business with.

Michael Rusnack
Hewlett-Packard Company
Disk Memory Division
11413 Chinden Blvd.
Boise, ID 83704
(208) 396-2054

Introduction

The *Small Computer Systems Interface* (SCSI) is rapidly gaining in popularity as the interface of choice for computer system developers. As a disk drive interface, the intent was for application on small computers. Over the years this interface has grown in popularity among system developers. Companies have pooled resources in the development of today's SCSI specification. Emerging from these standards, many workstation manufacturers adopted SCSI as the peripheral interface of choice. Most recently, it was introduced for use on the Multi-User (mini-computer) systems.

The use of this interface has many implications to the end user. This presentation discusses those implications by first presenting a short history from its inception to the present day. Next, this presentation will discuss terminology and features that define SCSI devices. Included in this discussion are how these features can impact the system implementation. Third, a review of device/system features, and their possible incompatibilities. Fourth, the SCSI interface is compared with other peripheral interfaces in an effort to expand the understanding of the reader. Finally, the presentation briefly covers future implementations of the SCSI interface.

A Short History

The interface standard finds its roots in the 1960s where it began as an IBM mainframe interface. IBM 360s had a byte-wide, parallel I/O bus that could do fast block transfers to and from peripheral devices. Then, this bus was called the Selector Channel, subsequently known as the IBM OEM-channel. After some political wrangling, this parallel I/O bus evolved in the ANSI committees as the intelligent peripheral interface (IPI). Workstation users may be familiar with this bus. At about the same time, Shugart Associates envisioned the need for a flexible I/O bus. The result of this work was known as the Shugart Associates system interface (SASI). When the proponents of SASI approached ANSI with the suggestion that this interface specification become adopted as a standard, they learned that they were competing with IPI, another high-level interface. Furthermore, they discovered that there was a third contender called the intelligent system interface (ISI). Rather than being a third contender for the high-end specification, SASI proponents opted to work with the standards committee that dealt with low-level interfaces, and called the new standard SCSI to set it apart from the others.

All of this work was taking place in the early 1980's. The interface was finally adopted as an Interface Standard by the ANSI committee in 1984. SCSI and its successor SCSI-2 have most of the capabilities of the IPI and a few it lacked. Whether it was politics, fate or just luck, even before the SCSI specification was finalized in 1986, it was more widely accepted than the IPI.

February 1982: ANSI X3T9.2 committee starts work on SCSI specification

Fall 1985 to Spring 1986: Common Command Set developed

1986-1990: SCSI-2 development

1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990

Mid-1980: Original SASI specification

April 1984: SCSI-1 Specification forwarded out of committee

June 1986: ANSI approves SCSI-1

October 1989 to February 1990: Comment period for SCSI-2

Despite its existence for over 9 years, relatively few people are aware of this interface. In an informal poll, MIS' with an average 15-20 years of computing experience, had not heard of SCSI until as recently as two years ago. This is a clear indication of how rapidly this interface is taking the industry by storm. The SCSI interface was initially intended for the Small Computer or PC user. As it has evolved, it eventually found use in the Workstation Market, and today is found on some mini-computer/servers. This is seemingly a big step for what was initially developed as a Small Computer Systems Interface.

An Overview

The SCSI bus comes in two different forms: *single-ended* SCSI, in which the signal's logic level is determined by a voltage relative to a common ground reference, whereas in *differential* SCSI, the signal is the potential difference between two wires. The signal transmission on the *differential* bus is much more robust and less susceptible to electrical noise.

The SCSI specification asserts that the *single-ended* cable can be no more than six meters long. This cable length not only includes the cable connection from the host to the drives, it includes any cables internal to the cabinets. A *differential* cable can be up to twenty-five meters long. *Differential* and *single-ended* devices may not be connected onto the same bus. Besides signal immunity and cable length, *differential* SCSI devices are capable of faster transfer rates than the SCSI-1 specification. The advantage of the cable length is many fold. The cable limitation applies not only to the external interconnect cable, but to the internal cabling also. A typical enclosure (HP tower or rack) has an internal length of 1 plus meters. This must be considered when connecting computers to devices and devices to devices.

The evolution of the SCSI-1 standard continued with the advent SCSI-2. SCSI-2 defines features that are very common in high performance environments. Some of these devices include:

- Complete compatibility with SCSI-1
- Transfer rates of 10MB/second on 8-bit SCSI bus
- 16- and 32-bit bus specifications that will ultimately offer 40 MB/second
- Device level error recovery to free the CPU for other tasks
- Command queuing will let intelligent devices improve their own performance

Delving deeper into the SCSI standards, one will learn that there are variations on these variations. Both the *single-ended* and *differential* buses are available in 8 and 16 bit data lines. These buses are often referred to as *narrow* and *wide*. Expanding further on the jargon of the interface, the interface was further defined in SCSI-2 to allow transfer rates at 10 MB/sec (when previously it was 5 MB/sec); this interface is often times referred to as *fast*. SCSI-2 also defined the differential bus, thus when discussing a 16 bit -- *differential* bus, it is often called *Fast-Wide*.

The variations of the buses and their characteristics are:

	Narrow /Wide	**SCSI-1/ SCSI-2**	**Cable Length**	**Total Devices**	**Transfer Rate**
Single-ended	N	SCSI I	6 m	7	5 MB/sec
	N	SCSI II	3 m	7	10 MB/sec
	W	SCSI II	25 m	15	20 MB/sec
Differential	N	SCSI II	25m	7	10 MB/sec
	W	SCSI II	25m	15	20 MB/sec

With the advent of *Fast* SCSI as defined in the SCSI-2 specification, problems surfaced while operating the single-ended interface. The physical limitations of hardware drivers in the computer, the target disk and the cable became a factor. This limitation was noted in the reliability of the data transmission. While attempting to transfer data at 10 MB/sec, using a 6m cable, the operations consistently failed. The failures were due to noise generated as a result of the fast transitions required by this new, higher transfer rate. This failure was not observed in the differential transmission, just the single-ended. To alleviate the problem, when operating at 10 MB/sec and using a single ended interface, a maximum of 3 meters of cable may be used.

The wave form below is a SPICE model of the signal. The jagged transitions cause false signals to the controllers, thus failing the operation.

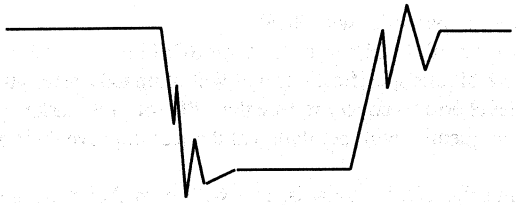


Figure 1 - SPICE - ACK wave form from AMP model

Hard disk drives are not the only SCSI devices available for use on systems with this interface. SCSI devices available today includes:

- Floppy disk drives
- Hard disk drives
- Tape drives
  - ¼ inch cartridge
  - 4-mm Digital Audio Tape (DAT)
  - 8-mm
- Optical drives
  - CD-ROM
  - Read/write
  - WORM
- Printers
- Scanners

An advantage of SCSI devices is the fact that multiple types can be connected to the same bus. Though there are restrictions, for the most part, the standard will allow the collection of disk drives, tapes and optical devices -- not only on the same bus, but in the same enclosure.

Though not completely, I have described the SCSI bus, along with some of its characteristics. To date, numerous papers and articles have been dedicated to the technical execution of SCSI as well as its impact on system development. If a more in-depth study is desired, I would suggest one or more of these articles.

SCSI -- Compatible or Maybe Not

SCSI devices are becoming more and more pervasive as a standard for interfacing different vendors' to a wide variety of peripheral devices. Chances are that if you own an HP or Sun workstation, a Macintosh, a NeXT cube, an Atari ST or an Amiga with a hard disk drive you may be already using SCSI. With these computer manufacturers and more adopting SCSI as the interface of choice for hard disk drives (and other peripherals), you might think that just about any disk drive would work on any computer system -- NOT SO. This lack of compatibility is despite the standards and carefully documented by ANSI in the SCSI-2 Standard and common command set (CCS). The implementation of the SCSI command set and its features are at the discretion of the developer.

This interpretation of the SCSI specification allows each developer of a system -- system interface to peripheral, has creative license. The interpretation allows the developer to design a system that is just slightly different from the next. This differentiation may offer an edge over a competitor's system. An example is *read-ahead cache.* This disk drive will read a "block A" as commanded by the host, then independent of host request will fill its buffer with the next several blocks. If the next read command from the host is for "block B" -- the data is immediately available.

Another example is a tuned system approach. The developer will oversee the integration and development of all pieces of the puzzle. In general, there are three major pieces to the system: host driver, system interface and peripheral. By using the products specified by the developer, the system is optimized. That is the best overall performance is achieved. Another reason to use the *recommended and supported* peripherals, you as an end-user is assured that the configurations are tested.

Remember, not all disk drives are the same. As a purchaser and user of these systems, you are often forced to make trade-off; balancing price with performance. Again caution is advised. HP and other makers of computing systems offer several flavors of their drives. In essence, all drives appear to work--so what is the difference?

- Performance -- it appears to work. Is that good enough?
- Bootable device -- does the system recognize this as a *boot device*?
- Feature set -- the system expects to negotiate for a feature and commands it.
- Connector -- does the drive have the correct I/O connector?
- Power fail support -- the drive completes sector writes during power fail.

These are just a few questions that could be asked. It is my personal experience that even within Hewlett Packard, the same 1.3 GByte disk drives achieve their personality from the firmware. The firmware in this or any other vendor's drive determines the characteristics and feature set. My best advise is to consult the systems sales and configuration guide -- otherwise **caveat emptier.**

SCSI versus Other Interfaces

Thus far, one might begin to wonder how the world ever survived without SCSI? Well, it has, and in many cases -- it continues. In the world of workstations, SCSI has been a standard interface for several years. Today, migration of this evolving interface is seen on mini-computers. Computing systems that in the past that have only used proprietary peripheral interfaces. In HP's case, HP-IB (parallel-copper wire) and more recently HP-FL (fiber channel) are two examples. Other companies like DEC and IBM offer multiple interfaces, including SCSI.

The reason SCSI is emerging as the interface standard on many computer systems is cost. The cost saving is realized by both the developer and end-user. The developer now has a standard to develop against. Of course, this is less expensive, with the savings being passed on to the buyer. If SCSI were so good then, why do companies continue to offer a second choice -- their proprietary interface?

As an R&D project manager, I had the unusual experience to partake in a performance comparison. SCSI and FL specifications were compared from the data sheet. The specification compared was -- *performance.* From the data sheet, 4- and 5 MBytes/sec transfer rates were noted for FL and SCSI interfaces respectively. With this data, it appeared a sure bet that SCSI drives would out perform the FL hardware. When the results were published, the opposite was observed. The FL drives bettered the SCSI consistently by 10 plus percent. The benchmark was performed on identical systems, except for the disk drive interface. The SCSI interface required more direct control by the CPU, whereas the *proprietary* interface operated much more independent of the CPU. This experiment showed that SCSI works; it may be simple and low cost. However, where speed and raw performance are necessary, it may not fit the entire bill.

Looking Ahead

SCSI-2 came about because several manufacturers wanted to increase the mandatory requirements of SCSI and define features for direct access devices. While maintaining compatibility with SCSI-1, SCSI-2 far exceeds the original standard. This coupled with the common command set (CCS), the SCSI standard was extended to all device types, adding caching commands, performance enhancement features and other worthwhile functions.

Two significant options offered in SCSI-2 was *wide* SCSI (up to 32 bits wide) and *fast* SCSI (synchronous transfers up to 10 MBytes/second). The combined effect of these two features alone could result burst transfers of 40 MBytes/second. *Fast-Wide* disk drives are expected to be available this Fall. These devices (16 bit wide) are capable of transferring at 20 MBytes/second. These devices will exceed the performance capabilities of the proprietary interface noted above.

Eventually, the SCSI standard will evolve to a fiber channel--capable of up to 100 MBytes/second. SCSI *fiber* is part of the SCSI-3 specification. It is not known when the first products will be available, however there is sufficient room for growth in performance and feature set with the SCSI-2 definition until then.

In Summary

Has SCSI met its promise and lived up to its early expectations? The initial goal was to offer multiple plug-and-play compatible types of peripherals on the same physical bus. Today it is the host adapter (SCSI interface for the computer) that is a considerable part of the equation. If the operating system drivers for that particular interface are developed the goal may be met. Another facet of the equation is the disk drive. Not all SCSI devices are created equal. Each has its own personality. The plug-and-play intention of the SCSI standard is most successful when using supported devices. Rather than gambling on loss of data, functionality or system down time, use the recommended peripherals.

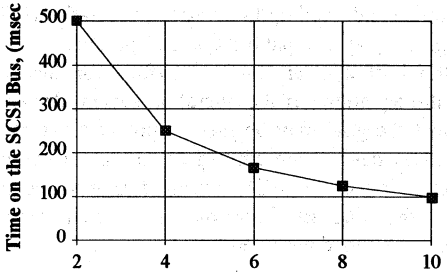
Earlier, I discussed the various attributes of the peripheral interface --*single-ended* and *differential.* In conclusion, I wanted to follow up on this discussion. The *single-ended* interface, by far the simplest and most versatile of the two has advantages and disadvantages. A real advantage is the connectivity. In the ideal plug-and-play world of SCSI, multiple device types may be housed on a single enclosure. A typical configuration might be:

TAPE BACK-UP
CD-ROM
MULTIPLE HARD DISK DRIVES

The flexibility of these devices affords the user to put up to seven devices in a single enclosure (and/or on a single bus). This reduces cost and increases reliability with the elimination of extra enclosures and power supplies. The price however is in the cable length constraint. The maximum length of cable from the host driver electronics to the final termination is six meters (or approximately 18 feet). When combining several devices/enclosures together, six meters is not very long.

Another short coming of *single-ended* SCSI is its transfer speed. This is the time it takes to transfer the data from the peripheral to the host or vice versa. This is important when systems are moving vast amounts of data back and forth. A *single-ended* device can burst transfer data at up to 5 MBytes/second.

From the following chart, it can be observed how transfer rate can affect performance. The faster the transfer rate, the less time spent on the data bus for a fixed amount of data moved. This performance bottleneck is analogous to traffic on a congested freeway. The faster the traffic moves, the more traffic that is allowed to pass, likewise with the SCSI bus. The faster interface --*differential* SCSI is faster, thus allowing more data to be exchanged.



As you may well have observed, the selection of a SCSI peripheral is no simple task. The application, hardware and software all must be brought together, with an understanding of how the system is to play together. This can only be accomplished through careful planning and development. Even though the*plug'n play* dreams of the first SCSI specification may not yet be realized, you the end user continues to benefit. Development is leveraged from device to device -- platform to platform. In general, you are offered storage devices at a lower cost and performance point that exceeds the rate projected just five years ago.

Conclusion

Now that you have experienced a non-technical overview of SCSI, what does it mean? By no way are you an expert. Engineers with technical degrees spend hours in seminars to cover the technical nuances of this evolving specification. The intent however is to provide you with a background; a greater understanding of an interface that will most likely be offered on your next new computer purchase or even offered as an upgrade in the future.

An Innovative Solution for Monitoring the Grouting Operation at Waddell Dam: A Case Study

Arden Mendenhall
Greg Neely
Michael Lemanski

United States Bureau of Reclamation
Arizona Projects Office
23636 N. 7th Street
Phoenix, AZ 85024

OVERVIEW

Electronic monitoring of dam foundation grouting operations has been done on a very limited basis. This may be due to initial startup costs and equipment capabilities in a harsh environment. The Bureau of Reclamation (BOR) has performed electronic grout monitoring on five dams since 1982. The systems employed have evolved from flow meters and pressure transducers with a simple strip chart recorder to a sophisticated telemetry and multi-user computer monitoring system. This system provides real-time monitoring and recording of the grouting process; printing of reports, summaries, statistical data; and plotting of the plan and profile drawings of the as-constructed results. The grout monitoring program for New Waddell Dam, located near Phoenix, is being monitored on a real-time basis by a grout monitoring system.

The original grout monitoring program was written in BASIC and used double linked-list data structures to store data. These structures allowed the system to search through one table and reach data in another table by using linked-list pointers incorporated within the structure. The system ran under HP BASIC/WS, a single-user environment, which often led to slow response and data deadlocks. The double linked-list data structures, which allowed for efficient computer search and storage capabilities, made updating by end-users a very tedious process. These conditions were among the reasons that brought about the upgrade of the grout monitoring system, which was completed in September of 1992.

The upgraded system allows for multi-user access to the program and uses an HP/ALLBASE data base for data storage. The system still exists in BASIC, but now uses BASIC subroutines. These subroutines use SYSTEM V pipes which in turn access a "C" routine that allows data communication between the main BASIC programs and the ALLBASE data base. System logic remains essentially unchanged, but the data update capability and multi-user environment make for a much friendlier system that is easier to use than the single-user system running under HP-WS. This paper will describe the process of foundation grouting and the evolution

of the system to its current state, with the emphasis on the most recently completed upgrades.

NEW WADDELL DAM

New Waddell Dam is located on the Agua Fria River, 35 miles (56 km) northwest of downtown Phoenix, Arizona. New Waddell Dam is the main storage reservoir of the Central Arizona Project, a 330 mile long canal that will bring water from the Colorado River to Phoenix and Tucson, Arizona as well as farming communities and Indian reservations between the Colorado River and Tucson. The dam is an embankment dam, and creates a reservoir with a conservation storage capacity of 812,100 acre-feet (an acre foot is the amount of water required to cover one acre with one foot of water, approximately 326,000 gallons) with a surface area of 9,970 acres.

The conditions of the dam foundation and an east-west trending ridge on the right abutment are such that grouting is required to produce an impermeable barrier to the water that will make up Lake Pleasant, the lake that will back up behind the dam. Both of these areas are made up of conglomerate and volcanic rock of tuff and andecite. The permeability rates for the materials can range from 0 to 200 feet per year (0 to .00002 cm/sec) up to as high as 320,000 feet per year (.31 cm/sec).

GROUTING PROCESS

The Bureau of Reclamation uses pressure grouting primarily to eliminate seepage and to reduce uplift pressures in the foundation beneath the structure. Grouting also increases the bearing strength and stability of the foundation rock by filling any voids or fissures that may be present, and the settlement of the structure may be reduced.

Pressure grouting, for the most part, is the process of injecting grout (slurry made up of neat cement and water) under pressure into holes drilled into the dam foundation for the purpose of sealing seams, cracks, joints, voids, or other defects through which water may travel.

Grouting involves the drilling of holes into the foundation of the dam, normally under the area covered by the impervious (clay) core. At New Waddell Dam, two types of holes were used to grout the foundation of the dam: blanket holes and curtain holes. The blanket holes were typically shallow holes ranging from 15 to 30 feet deep, spaced at ten foot intervals covering the entire area of the clay core of the dam. Curtain holes range from ten feet to 550 feet deep depending on what was needed to sufficiently treat the foundation and create an impenetrable grout curtain. The curtain holes were in a single row down the centerline of the dam, with extra rows added as necessary to achieve closure of the grout

curtain (a condition achieved when the holes will no longer accept grout). The Bureau of Reclamation normally drills the holes to depth, then tests and treats the holes in stages from the bottom to the top. The holes are tested with water under pressure for five minutes to see if they meet the criteria requiring grout. If the hole requires grouting, the Bureau will start out with a 5:1 water to cement ratio, by volume (five cubic feet of water to 1 bag of cement). Then, depending on the grout take of the hole being treated, the grout mix will be thickened by reducing the amount of water per bag of cement, until the hole no longer accepts grout (refusal). The pressure used in testing and grouting depends on the depth of the stage being treated. The Bureau uses one to one and half pounds per square inch pressure per foot of depth.

GROUT MONITORING EQUIPMENT

To successfully implement an electronic grout monitoring application, the BOR required electronic pressure transducers, flow meters and density meters. The pressure transducer is required to maintain water and cement pressure down the hole being worked on. Flow meters display liquid flow in gallons per minute. Since flow meters do not work well near zero value, a flow meter is used on both the supply and return lines. Return values are then subtracted from supply values to derive the actual down hole flow. Density meters monitor the water/cement mix that is being injected into a hole. The density meter together with the flow meters are used to calculate bags of cement injected and hourly rate of injection.

These three types of meters are essential to maintaining a communication between the field, where the actual grouting application is taking place, and the computer, where the storage, calculation and display of monitoring data occurs. These meters must be cleaned often and maintenance is essential to keeping correct readings.

TELEMETRY EQUIPMENT

Data is transmitted from the grout site to the computer by use of a telemetry system. Initially, data was sent by hardwiring the electronic meters to the computer. Extreme difficulties were encountered using this system, as cables would become entangled with other activities going on at the construction site. This system also limited the area that grout monitoring could take place, because of the logistics involved in the constant moving of equipment from one location to another.

The telemetry system has proved to be a much more efficient method of communication between the computer and the field. The system consists of Remote Terminal Units (RTUs) and a Central Terminal Unit (CTU). The RTUs consist of a microprocessor based

complex with firmware developed specifically for the main data acquisition system, which is the computer station. The RTUs collect the field data from the electronic meters for transmission to the CTU. The RTUs are located in the vicinity of the grouting operations so that hardwires are shortened and are in direct control of the grouting operators. A schematic of the system appears on the following page.

The CTU located at the monitoring station contains a program that receives the data from the RTUs and makes the data available to the master processing computer station. Computer monitors, printers and plotters at the central computer station provide readings of all system parameters on a periodic basis as well as tracking and time stamping of all changes of alarm states within the system as they occur. These readings are supplied to the system via a remote polling discipline.

The CTU can receive input data from up to 15 separate RTUs. Each RTU has 12 analog channels, allowing the RTU to transmit data for three grout pumping stations. Each station sends four pieces of information, pressure, flow in, flow out and density.

COMPUTER EQUIPMENT

The grout monitoring program runs on a Hewlett-Packard system that allows for data input, graphics display, report printing, and plotting outputs. The data acquisition portion of the system converts the incoming signals into numerical and graphical data and displays them on the color monitor. Graphical data consists of a flow rate curve, pressure curve, bag rate curve and a water-cement ratio curve. Numerical data displayed are port number (hole number), hole pressure, flow input, flow return, flow into the hole, density, water-cement ratio and grout take expressed in bags per hour. During the grouting or water test process, numerical data includes stage depth, cumulative cement take in bags and elapsed time of the water test. Only one set of curves can be monitored, but curve displays can be switched instantaneously to other ports which are simultaneously being grouted or tested. Numerical data from six ports can be monitored simultaneously.

Terminals are available to computer operators for data input. Operators are responsible for maintaining system calibrations, water test starts and stops, grouting starts and stops and report and plot requests. Operators are also in constant contact via radio to field inspectors for the purpose of relaying important information received from the field to the inspector.

Figure 1 shows an overview of the telemetry system and computer equipment.

ORIGINAL GROUT MONITORING HARDWARE SCHEME WITH FIELD SETUP AND TELEMETRY

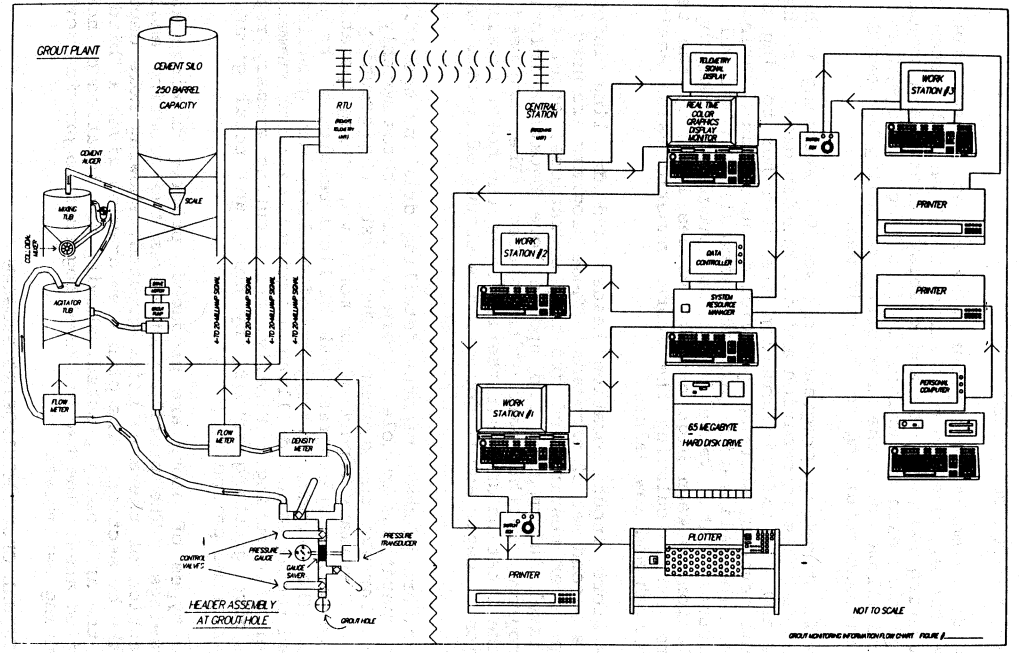


Figure 1

THE GROUT MONITORING SYSTEM

The grout monitoring program is a set of seventeen BASIC programs that are used to maintain calibration and system data, input grouting data, create and display reports and graphs, and display graphic data and update the data base. The system is menu driven and allows users the flexibility of moving between different options without returning to the main menu.

The first set of programs are used for entering in hardware defaults, dam axis information and calibration data. The hardware defaults and dam axis information are used for routing data to peripheral devices, setting up device defaults and creating report headings. The calibration program recalculates field values into their actual data values. This program works by using interpolation between high and low values input by the terminal operators and calculating the stored numeric value against the inputted high and low values.

Operators input location header and drilling information through a second set of programs. These programs are used to start and stop water testing and grouting operations, enter comments, enter travel information and update hole backfill information. These programs are the heart of the system. They are used to define the location and characteristics of the hole, to allow data collection to begin and to document the status of operations as it proceeds.

Operators produce reports detailing drilling operations and grouting operations by hole. There are also three monthly reports available through the system. The first, the grouting program report, is a summary report of the monthly grouting operations and is used in reviewing progress and the final results of the grouting program. The second, the report of drilling and grouting, is a condensed report of all drilling and grouting reports summarized in the order of grouting. Finally, the grouting summary report is used to give grouting totals, by hole, and total grout placed during the time period.

Two types of plots are produced, profile drawings and plan view drawings. The profile plot gives the grouting engineer a cross-section of the area being grouted, showing hole depths and grout take by section. The plan view drawing gives a "bird's eye" view of the area being grouted, showing the location of holes for a specified area. The two are used together to provide work plans and grouting estimates.

THE ORIGINAL GROUT MONITORING PROGRAM

The original grout monitoring program had three main deficiencies. First, the system was executed on older equipment

which ran in single-user mode. Next, the data was stored in linked-list data structures which were not reliable. Lastly, when the data structures needed to be corrected manually, the users had great difficulty finding the record in error.

The original system ran on Hewlett-Packard Workstations without multi-user capabilities. Monthly reports were often requested while real-time processing was going on. This significantly impacted the system, since resources were forced to be shared between the requested report and the current grouting activities. The first step to upgrading the system was to install new equipment which could handle multi-processing activities.

Figure 1 shows an overview of the original computer equipment. HP9000 Series 200 Workstations were connected to a Shared Resource Manager (SRM) running HP BASIC 3.0. The SRM server was an HP 9000/220 and the color display monitor was an HP 9000/236. Workstations one through three were HP 9000/216's.

Double linked-list data structures were the original means of storing the incoming data. Each structure was linked to at least one other structure through the linked-list. When the system experienced a power fluctuation or was impacted in some other way, the pointers that made up the linked-list interface would be corrupted, meaning that data would be lost to the system at the point in time that the power fluctuation occurred. The best method for improving the reliability of the collected data was to install a database to hold the data and maintain its own pointers.

Figure 2 illustrates the connectivity between the series of double linked-list data structures. This figure illustrates the use of pointers to link records of the same structure as well as to link records of different structures.

When the data pointers became corrupted, the main users had the task of restoring the pointers to their original state and connecting up the lost data through the corrected pointers. Often, this meant tracing through up to four data sets, record by record via the linked list, until the corrupted record could be found. An example of the pointer tracing required is illustrated on the following page. This job could take hours to conclude because of how involved and interconnected all the pointers were. Through the use of a query language, the time and effort required to make corrections could be greatly reduced and made much easier.

THE NEW GROUT MONITORING SYSTEM

The process of upgrading the Grout Monitoring Program began with the purchase and installation of new hardware. An HP 9000/385 was set up as the main CPU, and was connected to a 330 megabyte hard drive and a 660 megabyte hard drive. The HP 9000/385 allowed

DOUBLE LINKED-LIST DATA STRUCTURES AND THE POINTER-LINK PROCESS

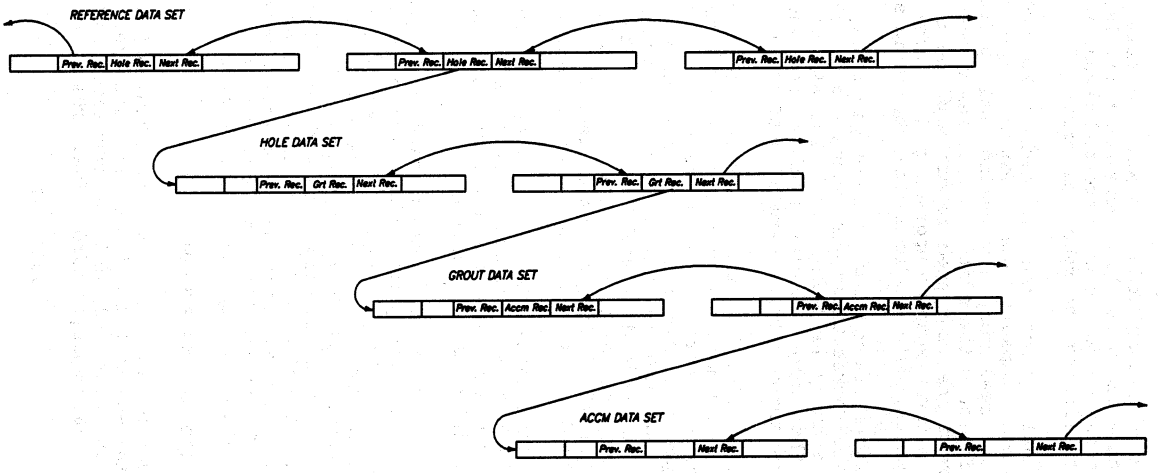


Figure 2

the system to be used in multi-processing mode, meaning that users could run the system in normal, day-to-day fashion with little impact when monthly reports or graphs were requested. Peripheral devices include four HP C1701 X-terminals with two megabytes of memory each, two 2379 printers, a 7570 plotter, a 9144 tape drive, a DAT Tape drive and a CD-ROM compact disk reader. The disk space more than doubles the previous storage capacity, and the multi-processing environment allows both printers and the plotter to be used simultaneously.

Figure 3 shows the new hardware scheme. The HP 9000/385 Workstation running HP-UX 8.0 is the heart of the system. Four C1701 Monochrome X-stations act as user input stations. Each X-station, as well as the 385 server, runs a BASIC/UX process, which places a significant processing load on the server. The 2 gigabyte DAT drive acts as a backup device and the CD-ROM reader updates software and reads on-line documentation.

The next step towards upgrading the system began with fixing the lost pointer problems associated with the double linked-list structures. It was clear that an industry standard database with a high degree of reliability was necessary and would replace the double linked-list data structures. ALLBASE/SQL was chosen. The existing software was written completely in HP BASIC 3.0 for the HP 9000/200 workstations. To minimize the code rewrite, as much of the original software as possible was preserved. In 1989, when the decision was made, HP BASIC/UX and ALLBASE/SQL running under HP-UX were chosen. HP BASIC/UX would allow the screen graphics and report generation routines to be retained.

HP-UX's networking would allow additional workstations to be connected via the Local Area Network (LAN), and also enable collected data to be sent to other sites for analysis. The job could be broken down into three parts:

1) Extract the input/output portion of the existing BASIC code and re-write it in ALLBASE/SQL.

2) Develop a scheme of communication between BASIC/UX and ALLBASE/SQL.

3) Convert the common BASIC 3.0 code to BASIC/UX.

Figure 4 shows the basic components of the new software design. The BASIC/UX programs are actually a collection of 17 independent routines that perform tasks ranging from data acquisition from the telemetry system to report and plot generation. User defined softkeys control which routine is executed.

Communication between the BASIC programs and the database

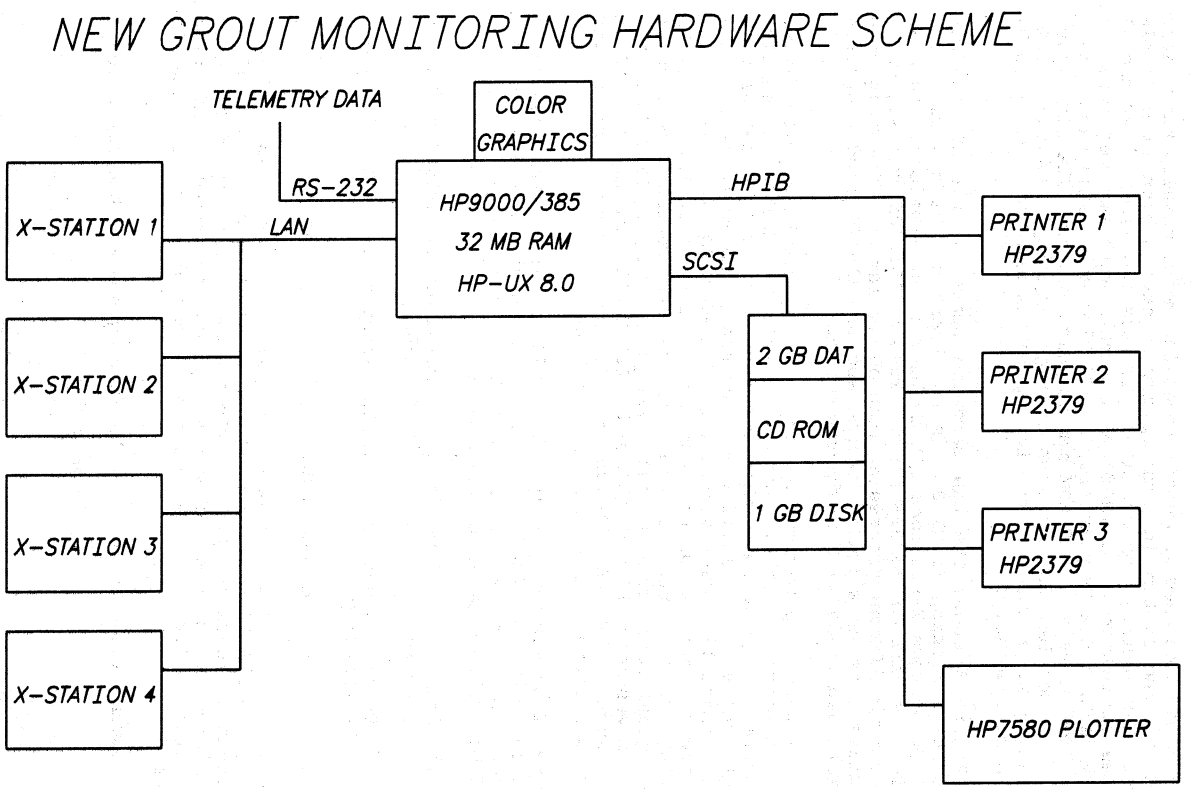

NEW GROUT MONITORING HARDWARE SCHEME
TELEMETRY DATA
COLOR GRAPHICS
RS-232
HP9000/385
32 MB RAM
HP-UX 8.0
HPIB
LAN
SCSI
X-STATION 1
X-STATION 2
X-STATION 3
X-STATION 4
2 GB DAT
CD ROM
1 GB DISK
PRINTER 1 HP2379
PRINTER 2 HP2379
PRINTER 3 HP2379
HP7580 PLOTTER

Figure 3

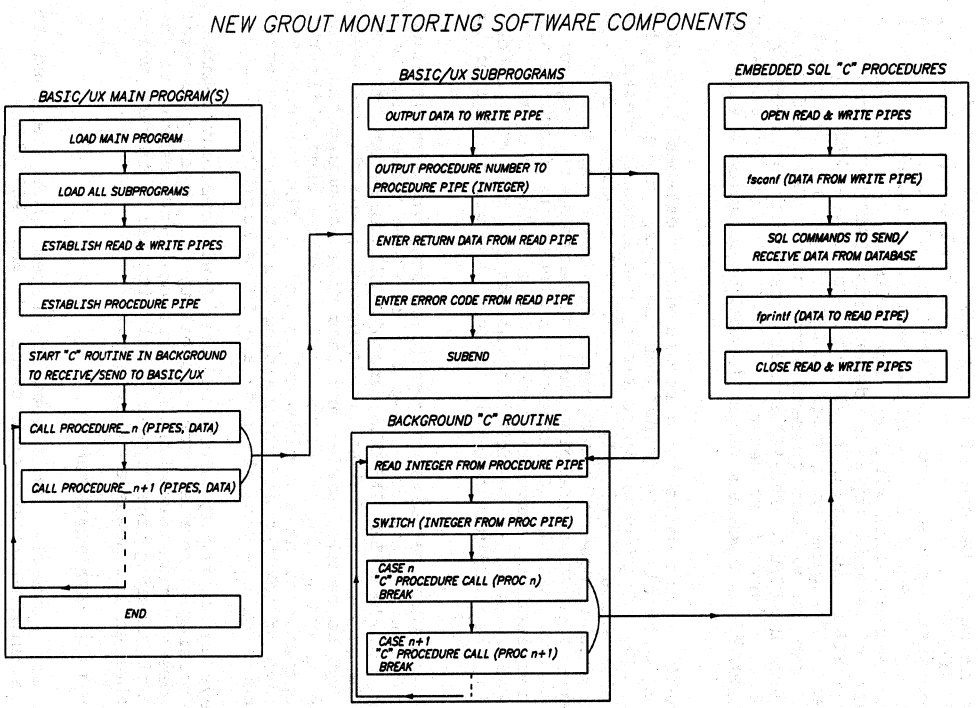

NEW GROUT MONITORING SOFTWARE COMPONENTS
BASIC/UX MAIN PROGRAM(S)
LOAD MAIN PROGRAM
LOAD ALL SUBPROGRAMS
ESTABLISH READ & WRITE PIPES
ESTABLISH PROCEDURE PIPE
START "C" ROUTINE IN BACKGROUND TO RECEIVE/SEND TO BASIC/UX
CALL PROCEDURE_n (PIPES, DATA)
CALL PROCEDURE_n+1 (PIPES, DATA)
END
BASIC/UX SUBPROGRAMS
OUTPUT DATA TO WRITE PIPE
OUTPUT PROCEDURE NUMBER TO PROCEDURE PIPE (INTEGER)
ENTER RETURN DATA FROM READ PIPE
ENTER ERROR CODE FROM READ PIPE
SUBEND
BACKGROUND "C" ROUTINE
READ INTEGER FROM PROCEDURE PIPE
SWITCH (INTEGER FROM PROC PIPE)
CASE n
"C" PROCEDURE CALL (PROC n)
BREAK
CASE n+1
"C" PROCEDURE CALL (PROC n+1)
BREAK
EMBEDDED SQL "C" PROCEDURES
OPEN READ & WRITE PIPES
fsconf (DATA FROM WRITE PIPE)
SQL COMMANDS TO SEND/ RECEIVE DATA FROM DATABASE
fprintf (DATA TO READ PIPE)
CLOSE READ & WRITE PIPES

Figure 4

takes place through the use of HP-UX SYSTEM V Pipes and a "C" routine. BASIX/UX can not directly access ALLBASE/SQL; however, BASIC subroutines written and compiled in "C" can. Compiled subroutines or "C Subs" in BASIC/UX require a large amount of overhead and recompilation with each new revision of BASIC/UX. In the interest of simplicity and decoupling BASIC/UX from the database, a SYSTEM V is used to bridge the gap between BASIC/UX and the database. The pipe is a simple FIFO (first in, first out) device that can be written to/read from with OUTPUT and ENTER statements from BASIC/UX. The "C" routine reads/writes to BASIC/UX with fscanf and fprintf statements. The "C" routine contains embedded ALLBASE/SQL commands which directly communicate with the database.

The BASIC routine waits for a return when a database activity is requested, effectively "waiting" for the background process to do its work. The "C" process, in contrast, continuously checks the pipe for information. The only time the "C" process halts its checking is when a request is received. At this time, the "C" process suspends checking the pipe until it performs the requested function. As soon as the "C" routine completes its function and sends any required information back to the BASIC routine, the process of continuous pipe checking begins again. This "C" process will run in the background until halted by grout monitoring personnel.

On the occasion that data corruption occurs, grout monitoring personnel can now use SQL to access the database tables and make corrections. SQL DELETES, INSERTS, and UPDATES can be used to make changes to large amounts of data simultaneously, and the end users find using SQL to be much easier and less time consuming than the old process of correcting pointers. SQL can be used while the system is operational at a minimum of impact on the system. The entire process of loading and maintaining the database uses only about one tenth of the time the old data files required.

The system currently runs in the X-windows environment, allowing grouting personnel the capability of opening several processing areas simultaneously. This option allows personnel the capability of running the system real-time as intended while pursuing other ad hoc needs as they occur.

DETAILS OF PIPE DATA TRANSFER

There are actually three separate pipes that are used to maintain the information bridge between "C" and BASIC/UX:

1) Write Pipe - BASIC/UX writes to this pipe and "C" reads it.

2) Read Pipe - BASIC/UX reads from this pipe and "C" writes to it.

3) Procedure Pipe - BASIC/UX sends an integer to this pipe corresponding to a specific procedure or task to be performed. The "C" routine reads this value and branches to the appropriate procedure.

The terms "read" and "write" always refer to BASIC/UX's perspective.

DETAILS OF THE NEW GROUT MONITORING SOFTWARE

As mentioned earlier, the BASIC/UX main programs consist of 17 separate programs that each perform a different function. These programs are loaded through softkey choices. BASIC/UX performs the screen graphics routines and all calculations on the telemetry data from the field. When data needs to be sent/received to/from the database, a "call" is executed with the appropriate variable list. The BASIC/UX subprograms handle the actual database transfer. Each database request from BASIC (update a row in the GROUT table, for example) invokes a specific "C" procedure containing the embedded SQL commands to perform the task. The following sequence outlines a BASIC/UX request to write to the database. The specific example illustrates the calls made when inserting a row into the GROUT table.

1) BASIC/UX makes a call to insert a row in the GROUT table
CALL I_grout_1(@Rp,@Wp,@Pp,H$,G_loc$,(more data))

Here, @Rp, @Wp, and @Pp refer to the read pipe, write pipe and procedure pipe, respectively.

2) Subprogram I_grout_1 receives the parameter list and does the following:

* Output the data to the write pipe. The data will remain in the pipe until it is read from the i_grout_1 procedure containing embedded SQL commands. (see figure 5)

* Output an integer to the procedure pipe. This is read by the background "C" routine in Figure 5 and in turn causes a branch to the i_grout_1 "C" procedure.

* Procedure i_grout_1 opens the read and write pipes, reads the data from the write pipe, and then sends the data to the database. A zero value is sent back to BASIC/UX to indicate the transfer was successful.

GROUT MONITORING PROGRAM SYSTEM ORGANIZATION

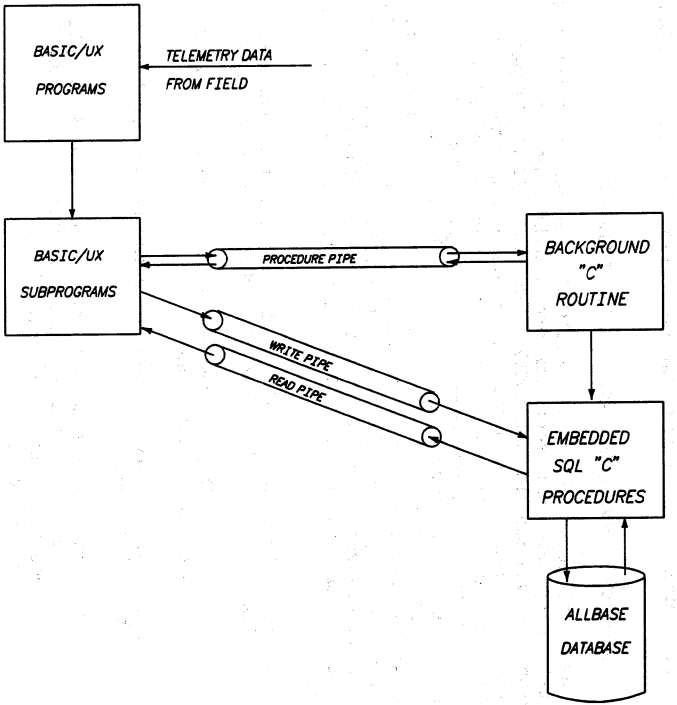


Figure 5

* Control is returned to BASIC/UX for other tasks.

FUTURE DEVELOPMENT

Part of the future development of the grout monitoring system includes interfacing the HP hardware with PC's through the LAN connection. By connecting one or two PC's to the system, the end user has the capability to moving HP graphics language programs into a PC based CAD system, for additional processing. Reports generated by the grout monitoring system can be imported into a PC based word processor for documentation purposes. The interfaces between the HP hardware and a PC are as numerous as the software products available to PC's.

The system can be connected to additional color graphics monitors or color printer/plotters to allow for color graphics dumping. Currently, the end user is limited to looking at telemetry data for only six holes at a time. By adding a second color graphics monitor, and making a slight program change, the user can view up to 12 telemetry data readings at one time. This is a significant improvement, since the end users may have more than four ports unused between holes being monitored,and currently must page between the active ports to view incoming telemetry data. The color printer/plotter would allow for a graphics dump of the current picture displayed. Currently, the end user has no graphics dumping capability.

Finally, the use of expert systems capabilities may be available to the grout monitoring program. In this situation, the system would be enhanced to not only read in readings from the field, but make corrections to the drive motor and flow meters as required by the current situation. Currently, if pressure or flow drop, operators must call to the field to inform the field inspectors that an unacceptable condition is occurring. Through the use of expert system enhancements, the system would be self-correcting.

CONCLUSION

The New Grout Monitoring Program allows end-users to work in a multi-processing environment, with no change to the operating procedures that the old system had. It uses an ALLBASE database, almost eliminating data corruption errors, and makes correction and maintenance a much simpler process. The new system contains more space for data, has a stronger data storage process and contains many more alternatives to running and using the hardware. The system also provides an interface between BASIC and ALLBASE, an interface rarely used, that will allow future developers more opportunities for system development and coding options. In conclusion, the New Grout Monitoring Program not only gives grouting personnel at the Waddell Dam a tool for more effective

operation, but engineers and programmers through out the world a flexible software interface that may satisfy many future needs.

Paper No.: 7027
System Administration - MPE to HP-UX
Ann McDermott
Hewlett Packard Division
Information Builders, Inc.
1250 Broadway
New York, NY 10001
212-736-4433

You are an experienced MPE administrator. Someone in management decides that we are now hopping on the bandwagon of Open Systems - all of a sudden into the middle of your controllable, familiar world someone drops an HP-UX box and says "Administer it." You're experienced! You say sure I can do it, then all of a sudden the manuals start arriving and you realize that what was once a nice, neat, orderly world is now looking pretty overwhelming and alien in light of the three-volume set of commands. Where do you start to find the information you need to run this system? Luckily the world of a System Administrator is fairly uniform across all platforms and HP-UX Manuals are very well written. This paper is designed for the experienced MPE administrator who will be or is now responsible for managing an HP-UX system. We will take various areas of responsiblity and compare the commands and procedures in both MPE and HP-UX. This is not meant to be a comprehensive guide but rather a brief overview of some of the more important and frequent tasks we are called upon to perform. Each command or name of an important part of the system will be printed in ***bold italic***.

The areas we will concentrate on are:

- backup and restore
- disk management
- IPLing the system
- process management
- user management
- upgrading the operating system and installing patches.

HP-UX does provide a user friendly interface called ***sam*** for some SA tasks but it doesn't cover them all. It is helpful to understand what ***sam*** is doing and to know how to perform those tasks from the command line yourself. We will touch on whether or not ***sam*** handles a particular task or function.

Let's start by briefly pointing out the differences in basic syntax between HP-UX and MPE. In MPE, each command is usually a word in the English language followed by options delimited by a semicolon, such as

showproc;job=@.

Someone with a knowledge of the computers and the operating system can glean what the command means just by reading it–show all the processes running on the system.

In HP-UX however, the equivalent task is performed with the following syntax:

ps -ef

It is not automatically apparent to the user what this command means or what it is doing. This is true for many HP-UX commands and to compound the difficulty there are very often several commands which can be used to accomplish the same end. We will try to focus here on giving the most commonly used methods. However, the beauty of HP-UX is that once you have an understanding of the syntax and commands it can be much faster, much more creative and sometimes more fun to work with.

Each HP-UX command has many different options available which are usually implemented using the '-' symbol followed by a letter or number. The meaning of these options can be found in the HP-UX help facility called ***man***. Type ***man*** and the name of the command you want help with and the system will display the entry from above referenced three-volume manual set. If you are reading the actual manual the System Administrator commands are listed in Volume 3, User Commands are Volume 1 and C Programming Commands in Volume 2. For quick reference, we recommend you print out the man pages for the commands you use most frequently using the ***lp*** command (ex: ***man ps|lp***) and create a personal notebook.

BACKUP and RESTORE

Backup and restore in HP-UX are not as simple and straightforward as MPE users are accustomed to. However, once you are familiar with the different methods of backup and it has been decided by your site which method is best suited to your needs backup can be painlessly automated.

In MPE to do a full system backup the command is:

file t;dev=7
store @.@.@;*t;show;directory

In HP-UX the same command can be done in a few different ways. See the following examples:

find / -print | cpio -ovBcd > /dev/rmt/0m

dump 0undf 6250 /dev/rmt/0h /u2

/etc/fbackup -0 -uv -g /usr/sam/config/br/grapha17336 -l /usr/sam/log/br_index.full -c /usr/sam/config/br/fbackup_config -d /usr/adm/fbackupfiles/dates -f /dev/rmt/c201d3m } 2>&1

tar cvf /dev/rmt/0m

Below is listed a brief summary of each form of backup and the pros and cons of using it.

cpio - ***cpio*** is a UNIX standard method of copying files. It is transferrable to all non-HP UNIX systems. However, when doing recovery on a tape made with ***cpio*** it can be very slow because it reads each byte of data on the tape. When used for backup it is combined with the ***find*** command to capture the desired data using the | symbol (see above). You must also use the ***find*** command if you wish to use this method for incrementals.

dump/ - rdump ***dump/rdump*** is also a UNIX standard. It allows you to do incremental backups by writing to a file entitled ***/etc/dumpdates*** which records the last level of backup done. ***Dump*** only allows you to backup one file system at a time.

fbackup/- frecover ***fbackup/frecover*** is an HP-UX proprietary method of copying files. This is the method ***sam*** uses. It is much faster than ***cpio*** on recovery. The ***sam*** interface is very straightforward and easy to use. We recommend this method.

tar - ***tar*** is also standard UNIX. It is the method of copying files used by most third party software vendors. It gives you the capability of compacting files into one big file. However, this method is not recommended for system backups as it does not allow you to cross tapes volumes.

Backup can be automated by placing the script of the method you choose in a file and using the ***cron*** facility.

Note of caution: Because there are a variety of ways to copy files users may choose different methods. It is therefore imperative in HP-UX that all tapes are not only labeled with the date and the data contents but the method of backup used. Otherwise, you may be asked to restore a tape and not know which method was used create it.

DISK SPACE MANAGEMENT

One of the most common tasks of an SA on any system is to monitor and manage disk space. In MPE this is done by issuing the ***discfree*** command with its various options (see ***discfree e*** below) or by running the ***report*** command against groups and accounts to see how many sectors they occupy.

```
DISCFREE A.01.03 Copyright (C) Hewlett-Packard 1989.  All rights reserved.
                        FRI, JUN 18, 1993,  4:46 PM

            |    Configured      |       In Use       |     Available      |
------------+--------------------+--------------------+--------------------+

TOTALS :
 Device     |   16400512         |   15227536 ( 93%)  |    1172976 (  7%)  |
 Permanent  |   16021040 ( 98%)  |   15031600 ( 92%)  |     989440 (  6%)  |
 Transient  |   14726368 ( 90%)  |     195936 (  1%)  |    1172976 (  7%)  |


ACCOUNT          FILESPACE-SECTORS      CPU-SECONDS     CONNECT-MINUTES
   /GROUP          COUNT     LIMIT      COUNT    LIMIT     COUNT    LIMIT
FOC67             1187872        **      60731       **     13966       **
FOCBETA            304848        **       2366       **      2271       **
FOCDEMO            207136        **     275089       **    130048       **
FOCUS              678480        **    2183237       **     53202       **
```

Note of Disk Configuration: In MPE all disk are dealt with through ***SYSGEN*** and ***VOLUTIL.*** In HP-UX all devices are addressed through files which can be found in the ***/dev*** directory of the ***root*** or / file system. Files for disks are located in the ***/dsk*** and ***/rdsk*** directory. Files for tape drives are in the ***/rmt*** directory. This is covered extensively in the System Administrator Manual. These device files are then mounted to a directory you name from the root file system. Disks may be added through ***sam***. ***Sam*** runs ***mediainit*** if necessary and then ***newfs***. The entries for the ***device file*** and mount points are added to the ***/etc/checklist*** file. The reference information for ***newfs*** is located in ***/etc/disktab***.

In HP-UX the command that corresponds to ***discfree*** is ***bdf*** combined with ***du***. The output can be seen below:

```
# bdf
Filesystem          kbytes    used   avail capacity Mounted on
/dev/dsk/6s0        313742  269651   12716     95%   /
/dev/dsk/c41d5s0   1277886  950790  199307     83%   /u5
/dev/dsk/c41d0s0   1277886  894766  255331     78%   /u4
/dev/dsk/c41d6s0   1277886 1135183   14914     99%   /u3
/dev/dsk/5s0        386928  215001  133234     62%   /u2
```

```
# du -s *
2         /
2         /CD_ROM
2         /Mail
4384      /SYSBCKUP
8690      /bin
86        /dev
45596     /etc
4384      /hp-ux
16338     /lib
16        /lost+found
9494      /system
2         /tmp
430002    /u2
2068824   /u3
1789532   /u4
1899908   /u5
224       /users
454226    /usr
```

Unlike MPE, the system in HP-UX is split into various partitions called file systems which are in essense disk drives. Space analysis is therefore done on a file-system-by-file-system basis. You then need to analyze the directories under each file system.

The ***du*** command which measures in 512k blocks, can first be used for global analysis. As the larger directories are determined a closer analysis can be done using either the ***du*** or the ***ll*** command. ***ll*** is a long listing of the contents of a particular directory, it measures the number of bytes in a file. However, it is only useful for measuring file sizes not directories. The entries you see for directories indicate the size of the entry for that directory in the inode table and will grow with the number of files in the directory. This is not the size of the files within that directory. In other words, a directory such as ***/dev/rmt*** may say 1024 when you run ***ll*** from the ***/dev*** directory but if you run ***du -s*** from there you may see that this directory is huge. A very common occurrence in this case is a user who inadvertently created a file rather than actually doing a backup by typing ***/dev/rmt/om*** rather than ***/dev/rmt/0m***. The system created a file called ***/dev/rmt/om*** containing the files to be copied. Imaginably, this can occupy large amounts of space on your root disk. Just by looking at the ***ll*** output from ***/dev*** you would not see this unless you ***cd*** into the ***/rmt*** directory where the file is located.

Below is listed a sample of the ***ll*** output from the root directory–notice how /u2, /u3, /u4 and /u5 are listed at 1024. These are mount point for 400 MB and 1.3 GB file systems. If you look in the above ***du*** example you will see the correct listing of space those file systems occupy.

```
# ll
total 9258
-rw-rw-r--   1 root     sys          535 May  4 14:49
-rw-------   1 root     sys          343 Jun 17 14:26 .Xauthority
drwx------   2 root     mail        1024 Mar 23 17:26 .elm
-rw-r--r--   1 root     sys           57 Feb 10  1992 .events
-rw-rw-rw-   1 root     sys           97 Apr 21 14:22 .glancerc
drwxrwx---   2 root     sys         1024 Jan 22 17:13 .lrom
-rw-rw-rw-   1 root     sys           17 Jan 22 16:39 .lromrc
-r--r--r--   1 bin      bin         8507 Dec 21 13:13 .profile
-rw-------   1 root     sys         2298 Jun 14 20:26 .sh_history
drwxr-xr-x   7 root     sys         1024 Jun 18 16:22 .vue
-r-xr-xr-x   1 root     sys         2415 Jan 19 15:50 .vueprofile
drwxr-xr-x   2 root     sys           24 Nov 17  1992 CD_ROM
drwx------   2 root     mail          24 Apr 27 11:25 Mail
-rwxr-xr-x   2 root     other    2232320 Jun 17 15:40 SYSBCKUP
drwxr-xr-x 109 125      qa          3072 Jun 17 09:18 batch_qa
drwxr-xr-x   3 root     root        3072 May 14 20:47 bin
drwxr-xr-x  13 root     sys         3072 Jun 18 14:37 dev
drwxr-xr-x  17 root     sys         5120 Jun 18 14:58 etc
-rwxr-xr-x   2 root     other    2232320 Jun 17 15:40 hp-ux
dr-xr-xr-x   4 bin      bin         1024 May 16 21:13 lib
drwxr-xr-x   2 root     root        8192 Jan 30  1992 lost+found
dr-xr-xr-x 165 bin      bin         4096 Jun 15 08:57 system
lrwxrwxr-x   1 root     sys            7 Nov  6  1992 tmp -> /u3/tmp
drwxr-xr-x  29 root     root        1024 Jun 10 14:07 u2
drwxr-xr-x  19 root     root        1024 Jun 18 16:10 u3
drwxrwxrwx  22 root     root        1024 Jun 18 16:13 u4
drwxr-xr-x  24 root     root        1024 Jun 16 14:38 u5
drwxr-xr-x   4 root     sys         1024 May 26 09:08 users
dr-xr-xr-x  35 bin      bin         1024 Jun  7 10:47 usr
```

At present there is no utility that we are aware of that measures the disk usage globally. However, the script below captures the picture of disk usage on a global scale. First it issues the ***date*** command measures the entire system using ***bdf*** and then runs ***du*** on each individual file system and sends the output to a file entitled ***diskout***. This can be customized for your site, printed and kept for analysis and tracking the history of disk usage.

```
use ()
{
date
bdf
du -s /*
du -s /u2/*
du -s /u3/*
du -s /u4/*
du -s /u5/*
}
use >diskout
```

Disk Space Tips: If your root file system is very full you can move directories such as */tmp* to another file system and link them using the ***ln*** or ***link*** command (see above). The */etc* directory may contain two files - ***wtmp*** and ***btmp***. These are log files used by the ***last*** and ***lastb*** commands which register the number of successful and unsuccessful logons. They tend to get very big. It is good to have this information online for security purposes but these files can be archived and recreated anytime using the ***touch*** command. You may want to either purge or store them off to tape periodically.

IPLING THE SYSTEM

In MPE the typical series of events for bringing the system down is:

1. Bring network down
2. Control A =shutdown
3. Control B CM> rs
4. ISL> start norecovery

As usual, there are a few ways to accomplish this in HP-UX. The two most commonly used are ***shutdown*** and ***reboot*** - both located in the */etc* directory. One very positive attribute of HP-UX is that unlike MPE it can be restarted from any terminal on the system, whereas in MPE it must be done from the console.

shutdown - Is the most graceful way to bring the system down. It uses ***kill -14*** to kill processes. If no options are stated, ***shutdown*** will issue, by default, a 60 second grace period during which users can log off. In addition, it will send a message indicating the system is coming down. You can also designate users other than root to use this command by entering their user id in the */etc/shutdown.allow* file. Here are examples of the syntax for ***shutdown***:

shutdown -h	will halt the system and allow you to cycle power
shutdown -r 30	will reboot the system with a 30 second grace period
shutdown	will bring the system into single user mode for maintenance purposes.

reboot - Reboot is a much quicker and meaner way of shutting the down the system. It uses ***kill -9*** and does not give a grace period. See syntax below:

reboot	immediately brings the system down and sends all messages to the console.

We recommend using */etc/shutdown* whenever possible.

You do not have to worry about bringing down the network. HP-UX will automatically bring it down and reset it with the system.

As in MPE, you can send a message to users using the ***wall*** command. To issue this command, type in your message and hit ***Ctrl D*** and the message will be sent. This is the equivalent of MPE's ***warn*** or ***tell***.

If you would like to see some of the procedures the system goes through in booting you can print out */etc/rc* which is run every time the system is reset.

PROCESS MANAGEMENT

In MPE to know what processes are running and which users logged on to the system we use ***showjob*** and ***showproc***. These display the current users and jobs on your system and the corresponding session or job number. To display this on a more detailed level you can do ***showproc;job=@*** and it will tell you the actual process numbers of each part of the session or job. If there is a problem with a session or job you first take the console and run ***abortjob #j/s[xxx]***.

```
showjob
JOBNUM   STATE IPRI JIN   JLIST    INTRODUCED   JOB NAME
#S196    EXEC         43   43       WED  4:10P  MGR.QA
#J78     EXEC         10S  124S     WED  6:36P  SRCH,ARU.ARU
#S291    EXEC         69   69       FRI  1:21P  QA.QA
#S308    EXEC         71   71       MON  9:45A  MGR.QA
#S422    EXEC        131   131      TUE  5:44P  MANAGER.SYS
#S429    EXEC         81   81       WED 11:48A  ARU.ARU
#S442    EXEC         84   84       WED  2:41P  YN.YN
#S445    EXEC         86   86       WED  2:56P  MANAGER.SYS

JOBNUM   STATE IPRI JIN   JLIST    SCHEDULED-INTRO    JOB NAME

#J121    SCHED    8  10S  LP        6/24/93 23:00     FULLBACK,MANAGER.SYS
```

Similarly in HP-UX this information can be displayed on a user or global scale. Users type ***ps*** to display all their processes. The global version of this command is ***ps -ef***, which will display every system process. To further qualify the information desired use ***ps -ef |grep [uid]*** or any character string you may be looking for such as ***ps -ef |grep ksh*** (see below) which will show all users who are running ***Korn shell***. With this command you will see user name, parent process ID, child process ID, the time the process started, which device file the user is accessing, how many CPU minutes they have used and a description of the process. As the system administrator in HP-UX or the owner of a process you can abort that process by using the ***kill*** command.

```
# ps -ef |grep ksh
  UID   PID  PPID       C    STIME TTY      TIME COMMAND
  stuart  1421  1391    0  Jun 25  ttypb    0:01 -ksh
    qaid 17662 17661    0 17:39:49 ttyp1    0:00 ksh
     iqk  6922  6921    0  Jun 28  ttysc    0:01 -ksh
     rid 26335 26334    0 10:36:43 ttys9    0:00 -ksh
    root 18457 18275    4 19:07:12 ttys0    0:00 grep ksh
     psh 16729 16727    0 17:07:23 ttysd    0:00 -ksh
```

Here if you type ***kill -9 1391*** stuart will be logged off the system. You have killed the parent process which spawned the process that runs his ***ksh*** or ***Korn shell***.

UPDATES AND PATCHES

If you have upgraded an operating system or installed patches in MPE you are familiar with the ***AUTOINST*** program and the procedures of gathering enough disk space on ***LDEV 1*** before you start.

We are glad to say that in HP-UX this is much easier and less painful. You do have to make sure you have sufficient space on the root file system before you start but HP-UX has incorporated tools into its ***update*** program which will warn you if you have insufficient disk space.

Updating an HP-UX operating system and installing new products are done through the ***/etc/update*** program. From this program you select which medium you want to use as a source. The default device is ***/dev/update.src*** which was the device for cartridge tape drives. Change this to ***/UPDATE_CDROM*** or ***/dev/rmt/0m*** for CD or tape respectively. This program is very user friendly and more or less self-explanatory.

Once the update is finished all components installed are referenced in the ***/system*** directory in the root file system. This directory lists all ***file-sets*** on your system. ***File-sets*** are how HP-UX stores parts of the operating system and additional products. Should you have the need to remove a ***file-set*** once it has been installed, HP-UX provides another rather friendly utility called ***rmfn*** which looks similar to ***update***. Be very careful when using it however, this program's sole purpose is to remove ***file-sets***. Although, it can be very helpful in viewing information on which file-sets are loaded on your system.

HP-UX patches are sent in a ***tar*** format. Depending on the type of patch, the installation procedure may vary but for the most part you restore the tape into the ***/tmp*** directory and run ***tar*** or ***sh*** against the files. This will often give you a ***README*** file which then tells you further information on how to install the patch. It may also require you to run ***/etc/update*** against the files you have just loaded. For most patch installations, unlike MPE, it is not necessary to take the system down.

Everytime you run ***/etc/update*** it creates a log in ***/tmp*** called ***update.log***.

USER MANAGEMENT

The MPE system has accounts with groups and users. To add a new account, group or user, issue ***newacct, newgroup*** or ***newuser***. Each user is assigned to a particular account and group and given specific capabilities. A users freedom on the system is dependent on the level of access given. Each user has a home account and group. To display or change information for a user we issue the ***listuser*** or ***altuser*** commands.

Similarly, in HP-UX each user is assigned a home directory and group. The user's freedom on the system is limited by the permissions assigned to the files and directories they work with. All the user information is contained in one file called ***/etc/passwd***. In this file there are seven fields delimited by : listed below. Additonally, in that file the user is assigned a group. The information for that group is contained in the ***/etc/group*** file. The user administration can all be done using ***sam***. However, it's good practice to know how to handle it manually. The above referenced files can be edited using any HP-UX editor. The three main components user administration are:

/etc/passwd:

```
root:qtP6BUvxjY0wc:0:3:Ann McDermott,hp 29,911:/:/bin/sh
daemon:*:1:5:*:/:/bin/sh
bin:*:2:2:*:/bin:/bin/sh
adm:*:4:4:*:/usr/adm:/bin/sh
uucp:*:5:3:*:/usr/spool/uucppublic:/usr/lib/uucp/uucico
lp:*:9:7:*:/usr/spool/lp:/bin/sh
lob:NwAwI4MRS88v.:16:200:Bill Lob,hp 29,411:/u4/lob:/bin/csh
hpdb:*:27:1:ALLBASE:/:/bin/sh
```

1. User ID
2. Password (encrypted)
3. User ID number
4. Group ID for that user
5. User Information which can be split into 4 different fields separated by commas: Name, Location, Office Phone and Home Phone. This information is used by several programs to display who is logged onto the system (such as ***who*** and ***finger***).
6. User Home Directory
7. Default Shell for user.

/etc/group

```
root::0:
other::1:
bin::2:
sys::3:
adm::4:
daemon::5:
mail::6:
lp::7:
users::20:demouser,amz,guest,ibiuk,joes,myg
programming::200:lob
```

1. Group ID
2. Password
3. Group ID number
4. List of users included in that Group. ***Sam*** will automatically handle this for you. If you decide to add users manually you must enter them to this file yourself.

The third component of every user is the ***.profile*** file for ***borne*** and ***korn shells*** or ***.login*** file for ***C shell***.

The default ***.profile*** for ***borne shell*** assigned by ***sam*** looks something like this.

```
# more .profile

# @(#) $Revision: 66.1 $

# Default user .profile file (/bin/sh initialization).

# Set up the terminal:
        eval ` tset -s -Q -m ':?hp' `
        stty erase "^H" kill "^U" intr "^C" eof "^D"
        stty hupcl ixon ixoff
        tabs

# Set up the search paths:
        PATH=$PATH:.:/usr/lib

# Set up the shell environment:
        set -u
        trap "echo 'logout'" 0

# Set up the shell variables:
        EDITOR=vi
        export EDITOR
```

Note of caution: In MPE the most powerful user is ***MANAGER.SYS*** or any user with **SM** capabilities. In HP-UX it is ***root*** or any user with ***UID 0***. HP-UX will allow user ***root*** unlimited access to all parts of the system (that includes being able to scratch the entire system!). It is, therefore, CRITICAL for security reasons that you limit the number of people who know the ***root password***!

SUMMARY

Although the hardware looks identical on both MPE and HP-UX, the operating systems vary greatly. One appears to be much more user friendly. The other can appear out right hostile. Just like learning a new language, it may take some getting used to but once you are familar with the commands and syntax you may actually enjoy running HP-UX.

Paper Number: 7028
The Management Consultant's Toolbox, Don't Be Left Without It

Leonard S. Block
The Apex Group
7151 Columbia Gateway Drive
Columbia, Maryland 21046

(410) 312-2653

Introduction

More and more companies are calling on Management consultants to help them sort through and explain the numerous options that are available in Information Systems Technology. With the advances in networking, database management, client/server computing, and office automation, finding the right combination of products is quickly becoming a top assignment for consultants.

What better way to explain these technologies then to showcase how you actually use these tools in your OWN work !

What are some of the common duties that a Management Consultant must perform ?

After performing many hours of user interviews, reviewing endless sets of documentation and analyzing company objectives, often the real work is just beginning. How does one organize and package all of this information that is presentable to a client. Often the biggest challenge is what information should be presented and in what format. As we all know, a poor presentation of a consultant's findings can negate many hours of hard work and excellent research that has taken place on the assignment.
Specific responsibilities that we face when packaging our results include:

* Writing Reports

* Preparing Graphical Presentations

* Providing an engagement status via Electronic Mail

* Transferring documents to others for review, editing and comments

* Sharing results

* Accessing mainframes or database servers to update Accounting or billing information

Time is Money

Often we underestimate the amount of time we need to perform the above tasks when developing cost proposals for engagements. Therefore it is extremely critical that a consultant have at their disposal the right tools to organize their findings quickly, accurately and effectively. Not having a complete and organized "Consultant's Toolbox" can erode a large amount of an engagement's profit.

Why not kill 2 birds with one byte....

Besides enhancing our profitability by having the right tools at our disposal when preparing documents, an even larger benefit can be obtained by having a readily available technology showcase. We are going to spend endless hours learning and integrating software and networking tools we need to package our findings. Why not capitalize on this investment of our time ? By creating a Management Consultant's toolbox, we can showcase and demonstrate to clients how PC systems integration really works. Integrating graphs with text, uploading data to servers for billing, creating slides from report excerpts, and using the power of LAN's to share data, is exactly the type of things that many of our clients are paying us to help them learn.

What better way to instruct a client than to show them first hand how system's integration can benefit their organization by showing them how you use it in your own work. This approach will afford you instant credibility that you "Practice What You Preach" , not to mention that you had to get the report or presentation done anyway. Showcasing your toolbox during the pre-sales cycle maybe that little edge you need over your competition to win the engagement.

How do you build your toolbox ?

I know that many programmers and application developers already have many of the tools that I am going to mention. The focus here is on the role of the Management Consultant and their toolbox. Often application developers and Management Consultants are one and the same. However, many times they are not. Many toolbox articles seem to focus on the technical programmer or analyst and their tools. Almost no mention is given to Management Consultants and their needs. As I have mentioned, a Management Consultant's toolbox can provide great dividends in cost savings, winning new engagements and lending credibility to existing engagements.

Since there are many excellent software, graphics, database and client/server tools on the market today, which ones should you include in your toolbox ? Remember, our tool box will be serving two functions. The first will allow us to report and package our work quickly and effectively and the second is to establish a showcase for "systems integration" for our clients. So the better a tool fits both objectives the more valuable it will be to you as a consultant. No matter what specific package you select, it should be able to contribute to these two main objectives.

Regardless of the specific package selected, it is important to have at least one package in certain functional areas. Each area plays a central role in "systems integration" that will be important to demonstrate to clients. Sooner or later you will need to perform these functions anyway in completing your assignments.

Toolbox Categories that should be included:

The following are the categories to include and some criteria to look for when selecting a specific package.

* Operating Systems

Needless to say, a graphical user interface (GUI) is essential in today's market. The majority of your clients will have DOS based machines with an extreme desire to utilize Windows and its power more effectively. Right away is a chance to showcase technology to clients. Windows 3.1 (current version at this time) has many features such as macros, startup folders, etc. that can be used as a starting point for your desktop. In addition, a Windows based environment will allow you to demonstrate in a limited way the concept of "multitasking". Many clients have a hard time understanding this concept on a PC workstation. As operating systems such as Windows NT and OS/2 become more popular, your showcase can then move with the technology to demonstrate truly "multi tasking" operating systems.

A note here about Unix. Unix is becoming the preferred platform for many company's servers and main computers. However, in my opinion it will be awhile before the "business community" will feel comfortable with Unix as the primary basis for their client PC workstations. Windows based client applications are still going to dominate the market place for years to come and therefore will still be among the things clients will be impressed with.

* Desktop Management

This is a very important area when demonstrating technology to clients and is very often overlooked. Most people say "I have Windows" and their done. There are several desktop management packages for Windows that can greatly enhance your ability to easily move from one application to another, automate repetitive tasks, print documents and access data. A small investment in such a package will pay off many times in getting work done quicker and showcasing some of technology "Magic". Remember, our point is not only to get our work done but to also maximize our effort by creating an arena from which to demonstrate technology.

* Networking

I am sure this area is at the very top of the list when your are consulting for your clients on how they can improve their information systems. Having a good understanding of how your own organization's network works can set an excellent example for clients on how they can maximize their network resources through improved file and print sharing. In addition, networking will provide the backbone for other showcase areas such as Terminal Emulation and Client Server computing.

* Document Creation

To many this area is thought of as just some Word Processing package. There lies a big mistake. Documents produced by consultants are often a collaboration of data, text and graphics from a variety of sources. Therefore it is critical one select a Word Processing package that can easily incorporate and maintain the diverse kinds of data that make up the "Compound Documents" that consultants must quickly produce. Many, many, many hours can be spent merging and editing graphs, clipart and bitmaps with written text. Making the right decision in selecting a package can save you a lot of pain and grief when reworking your report or presentation after it has been reviewed by a supervisor. In addition, by making the right choice in selecting a Word Processor, the need for additional graphics or spreadsheet programs may be eliminated.

* Image Capturing

Many consultants are faced with the challenge of incorporating snapshots or copies of screens into their reports. This is not the same as bringing in a pie or bar chart. Almost all of today's Word Processing and Graph packages provide for the integration of the two. Yet how about capturing screen or report layouts from applications ? By having a tool that can capture images quickly for incorporation into a document will give presentations a much higher degree of professional quality. It will be very impressive when a client asks how you incorporated their images in your reports without redrawing them.

* Graphic Manipulation

As everyone knows, life is not perfect and even the images, bitmaps, and clipart we capture may need some cleaning up. Often we want to add some text to the image or highlight one area of the drawing. Most of our drawings are captured in color but are printed in black and white. Contrast and brightness needs to be modified when converting from color to black and white. In order for visual objects to look impressive in reports, a little "doctoring" here and there. Having the right tool in our toolbox is the key to success.

* Terminal Emulation

With the move towards downsizing and client/server engines, more and more clients are in need of downloading transaction data. There is an overwhelming need for effective data transfer solutions from mini or mainframe computers to networks and or workstations for further analysis. This is one area where having a readily available demo ready for clients on a workstation can give a client a feeling that what they need done is possible. Actual simulation of this type of processing for clients has been one of my most successful techniques in my years of consulting.

* Client/Server Ability

One can not pick up a technical journal without seeing several articles, ads etc. for client/server computing. Yet how many clients really understand this concept ? You yourself are preaching it as the right solution but actually may never have experienced it hands on. I believe that one must be able to effectively demonstrate why this technology is considered the wave of the future. Client/server tools may not currently be helping you get any of your immediate work done at the present. If it is not helping you in packaging your presentations , find a way !!. The learning curve you will go through will once again pay off in increased credibility for clients in showing them that you are actually using what everyone else is just talking about.

So What do you recommend for each area....?

There are many good packages from which to choose from in each of the functional areas just described. The packages that I will mention are ones that I have chosen for my toolbox. I will explain what features in each of those packages led me to choose them. In some cases, software/hardware/networking packages were chosen for me based on my company's software, hardware and networking configurations. Please note that I am not endorsing any of these packages from a "marketing " standpoint rather that these packages have been effective for me. These packages are not only effective on their own but interact very well with each other.

* Operating Systems

DOS and Windows 3.1 . I think no further explanation is needed here.

* Desktop Management - NewWave from Hewlett-Packard

NewWave provides an extremely effective desktop management package. NewWave provides you with an Object Oriented desktop that has the same look and feel as an "Apple" or "MAC" computer. Items are not looked at files but as objects. This graphical drag and drop approach to document management makes it easy to manipulate your work. Object folders can easily be created to organize your work. Drag and drop printing is also provided. One of the main powers of NewWave is its Agent macro or scripting language. The Agent can perform routine or repetitive tasks. This is a very powerful tool for showcasing your technology via various automated Agent Tasks. NewWave is inexpensive and is well worth the price just for the Agent Task Language alone.

* Networking

This area will often be dictated by your organization's configuration. All of today's major networks (Novell, LAN Mgr. etc.) will be able to provide you with the necessary tools to showcase your toolbox.

* Document Creation

This is the one area where you have many choices. I have chosen AmiPro from Lotus. Besides having all the standard Word Processing features found in WordPerfect and others, it does an excellent job in incorporating and editing graphical requirements of a document. In my opinion, AmiPro provides the and easy and flexible avenue for incorporating and editing graphical objects within the text of a report. Its built in drawing package allows you to quickly add a picture or graph right where you want it to be. This feature has saved me hundreds of hours when editing the many compound documents I have had to create and revise over the years.

* Image Capturing

A shareware application called PaintShop allows you capture and edit screen images. PaintShop has proven invaluable when creating reports or manuals where actual replicas of reports and or screens were required. Besides its main function of taking snapshots of screens, PaintShop lets you dither color images from 256 colors, increase or decrease brightness, contrast or make a color image black and white. It also allows you to stretch or shrink an image to the desired fit that is needed in your document. One feature of Paintshop that can be extremely valuable to a consultant is its ability to work on graphic images in a variety of formats. This is very helpful when you are trying to merge a diverse set of data, text and graphs into a report or presentation for a client. Formats such as BMP,TIFF, PCX and WordPerfect's WPG are supported by PaintShop. PaintShop is a standard Windows application and takes up a very small amount of disk space and can be installed in 5 minutes.

* Graphic Manipulation

Almost all application developers using Windows are familiar with Paintbrush. Yet as a Management Consultant this standard Windows application can be extremely valuable when preparing your documents and presentations. Since it is included with Windows, no additional investment may be required in an outside graphics package. Paintbrush allows you change a bitmap's color areas while also providing you with a small toolbox itself for creating simple flowcharts, organization charts or other graphs using simple shapes, lines and arrows. As its name implies, Paintbrush's main strength lies in its ability to allow a user to change color combinations.

Paintbrush used in conjunction with a package like PaintShop can provide a one-two combination in graphic manipulation that is a well kept secret for many. Since Paintbrush can only work on BMP files, how do you edit an image in say a WPG WordPerfect format ?. You can open the graph up in PaintShop since it supports WPG files, save it as a BMP file and then edit its colors in Paintbrush. These two packages (PaintShop and Paintbrush) have been two of my biggest toolbox assets.

* Terminal Emulation

My organization possess both an HP3000 and HP9000 server. Therefore I need an emulator that will allow me to get to both. Even though I have and use Reflection (WRQ), I am strongly recommending you evaluate HP's AdvanceLink for your toolbox. Besides the standard emulation functions, AdvanceLink has a user friendly scripting language for file transfers. This feature is very important when showcasing your toolbox. AdvanceLink is a very viable alternative to Reflection especially if your consulting base of clients is HP oriented.

* Client/Server Technology

Here the recommendation is Microsoft Access for your toolbox. Access is less than a year old but already is gaining in its user base. Access is perfect for the Management Consultant due to its extremely easy way to build screens, reports, queries, graphs and last but not least databases. One can use its built in Relational Database Management System or connect to such popular engines such as Oracle, SQL/Server or Sybase. In a very short amount of time, one can built a small Access demonstration of how "Client/Server" technology works which will "wow" clients and spark interest which hopefully will result in additional consulting engagements.

Summary

Every professional needs a good set of tools to compete. Athletes have their equipment, musicians have their instruments, programmers have their editors and compilers and Management Consultant's need a set of tools as well in order to effectively compete. If you create an effective toolbox, you will be maximizing your time and effort in learning today's technology. Not only will you be able to get your work done quicker and more effectively, you will have created a built in showcase for new and prospective clients. You can prove to clients that in fact what you SAY can be done ACTUALLY can get done. If you follow the guidelines outlined in this paper, I am confident that you will be able to:

1) Win new clients
2) Receive new engagements from existing clients
3) Produce higher quality presentations and reports
4) INCREASE OVERALL PROFITABILITY OF YOUR BUSINESS !

You wouldn't leave home without your American Express card, well the same could be said for a Management Consultant's toolbox,

"Don't be left Without One"

Some Examples

The next 3 pages contain some small examples that were created with some of the tools mentioned in this report.

Example 1 - AmiPro embedded graph

This example shows a simple yet effective flowchart done with AmiPro's built in graphics editor. Note, no outside package was used. The graphics editor is easily invoked at the insertion point and the flow chart was created. By toggling back to text mode, the continuation of preparing the document is easily obtained.

Example 2 - Screen capture with PaintShop

This example shows a mainframe screen captured with Paintshop and saved as a BMP file. Notice, that the lettering does not show up very well against the black background. The letter's original color was green. In order to use this graph in a WordPerfect document we will "doctoring up" in Example 3.

Example 3 - Screen capture after cleanup with Paintbrush

Notice how with Paintbrush, some of the letters have been touched up (changed to white) to blend in against the black background. Now the bitmap could be incorporated into a report and when printed on a laser printer, would appear crisp and professional.

A low cost Client-Server application

Mark C. Halstead
Aircast, Inc.
92 River Rd.
Summit, NJ 07901
800-526-8785

Client-Server computing is cooperation between two or more computers. Each computer shares a part of the processing. Hopefully each computer is doing what it does best. Client-server computing does not require UNIX. Client-server does not require SQL. Client-server does not require that someone call your databases "relational". You could create a client-server application on a UNIX platform, using SQL commands against a relational database but you aren't required to. You can, in fact, create client-server applications with the tools you already have. In the example I'll discuss, a DOS PC is used for its ability to generate attractive graphics and act as a flexible data entry system. The HP3000 and TurboIMAGE are used to perform queries against a large database.

There is a growing list of tools available to help implement client-server computing. When looking at the different packages my first criteria was cost. There was no strong support for client-server at our company because few people knew its potential. I wanted to put together a demo application, but wanted to avoid wasting company money on a project that no one had really asked for. I evaluated three solutions to generate the PC application interface or front end. All of them need a terminal emulation package to provide an interface to the 3000. I chose Reflection 1 for Windows as the terminal emulation because of its strong command language, its ability to support dynamic data exchange, and because I already had it. The decision on the HP3000 side was easy. I chose QueryCalc because of its flexibility, power, and ease of use. I have worked with QueryCalc for several years and feel very comfortable with it. It is worth mentioning that had IMAGE/SQL been available when I wrote this application, I would have been tempted to use something like Forest & Trees to put this application together. But Forest & Trees would have still required a cash outlay that I wanted to avoid.

The first package I looked at to generate the front end was Visual Basic from Microsoft. Visual basic has a lot going for it. It's inexpensive -- under $200 and can generate stand-alone applications that I can distribute without buying more copies of the program. Visual basic is a very powerful and flexible programming language. The downside of Visual basic is that it is a real programming language and I am not a programmer. I began teaching myself to program with VB but realized that climbing the learning curve would take more time than I wanted to spend. I haven't given up on it, but I decided that learning VB was a long term personal growth project and not the best fit for this project.

I have been using NewWave from HP as my Windows desktop for some time and new that it was capable of generating my front end. The NewWave agent task language is a rich and powerful tool. Underneath the pretty interface NewWave is serious technology, it could do everything I needed to do. HP has priced NewWave very attractively but I would still need to buy a copy for every person using my application. I would also need to train

them to use the NewWave desktop. While I personally like it very much some of the people I showed it to didn't like it and didn't want to use it. I could make this project happen with NewWave but it wasn't the best solution.

The third option was Excel from Microsoft. Excel has some good application development tools built into it as well as being a very good spreadsheet and graphics package. The Excel macro language is powerful enough to do what I needed and isn't terribly difficult to learn. Best of all, it was already on the machines I wanted to use. I chose Excel for my interface.

How it works

The application I wrote takes information entered in dialog boxes through Excel and transfers that information to Reflection. The transfer is entirely in the background. The user never sees anything but the Excel screens. Reflection command files transfer the information to the HP3000. The HP3000 product QueryCalc uses the information from the Reflection command files to perform queries and return information. The Reflection command files then gather the information from QueryCalc and transmit it back to Excel. Excel then displays the information graphically. The whole process requires nothing from the user except to point, click, type a few characters, and click again.

The PC side

The PC side was by far the most difficult part of the whole project. Making Excel and Reflection work together with DDE took much more time than the work on the HP 3000 side. I found it helpful to break the project into parts, the interface, the Excel component, and the Reflection component. By working on only one component on a given day I was less likely to confuse myself with differences in command language and format. Luckily they are similar enough that it wasn't a major problem.

Designing the interface --

The first thing to remember when designing a user interface is the user. Who is going to use this application, what are they going to do with it, and how are they going to use it. Make a plan, sketch out the screens on paper and see how they flow. Once you know what you want to do you can set about doing it. Don't forget that you have to make your program or macros manipulate these screens, so treat yourself kindly and don't bite off more than you can chew.

Dynamic Data Exchange, a quick overview --

Dynamic data exchange (DDE) is one of the best kept secrets of Windows and Mac computing. Most people have heard of DDE and many have tried it with varying degrees of success, but few people really know it's potential. There is a good reason for this -- most of the information about DDE is buried in manuals that would give even a serious techie the heebie-jeebies. DDE is the way Windows programs communicate with each other. They are able to open a *conversation* on a variety of *topics*. It works much like cutting and pasting, but is automated and doesn't require the use of the Windows clipboard.

To begin a DDE conversation you need to supply the name of the program that is going to be the *destination*, and a *topic* for the conversation. The name for the destination program is going to be something like EXCEL, WINWORD, or R1WIN. The documentation for your software will tell you the application's DDE name. In Reflection for Windows you can specify the DDE name, this helps locate the target destination when there are more than one Reflection session active. The *destination* is the application you are going to send information to.

The topic of a DDE conversation is the target of the data you wish to send to or retrieve from. If you were using Excel as the destination the topic could be the name of an Excel file. In Reflection and most other applications you can open a system topic that can provide you with information about other topics that are available as well as the status of the application. In Reflection you may also open an RCL topic that allows you to place data in Reflection command language variables or run command files. The type of topic that is available depends on the destination. In Reflection a topic is usually a type of data. In Word or Excel the topic can be a type of data (like the system topic) but it is usually the name of a file that you wish to act upon.

Once an application and topic is selected we need to do something with them. The data transmitted between the applications is the *item*. The item can be text, numbers, or graphics. What is acceptable as an item depends on the source and destination applications. In Reflection it is generally the value of a variable. In Excel it may be the contents of a cell or a named range.

The differences in DDE implementation for different software packages make it hard to be more specific about how to use DDE. The basic building blocks of DDE are buried deep within Windows. The way each software developer accesses them is entirely their choice. This means that a statement that performs a DDE task is going to look slightly different in every application you use.

The Excel component

Microsoft Excel supplies the user interface. It provides dialog boxes, graphs and the macros to drive them. Figure 1 is what the user sees when the application is loaded. The buttons on the left activate macros that load dialog boxes for user input when appropriate, or simply perform tasks as in the case of the Update the Score button. Figure 2 shows the screen with the Update the Graph dialog box opened. The user may enter either an account ID number or the ID number of any invoice. Clicking "OK" or pressing the enter key accepts the data and continues processing the macro.

Figure 3 shows the code for the UpdateGraph macro. The first six lines of code control the action described in the previous paragraph. Figure 4 shows the code that defines the dialog box in figure 2. It isn't necessary to manually define the parameters for the dialog box. The Dialog editor that comes with Excel allows you to visually define and place fields in a frame. When the box is set-up to your satisfaction, copy it to the clipboard. When pasted to a macro page in Excel the "visual" dialog box you used in the dialog editor is converted to the code you see in figure 4. The textual representation that is displayed on the macro page allows you to fine tune the alignment and adjustment of the elements. The

range containing these values is named in a range so we can later reference them in the macro. The Excel manual and other Excel references do a good job of explaining the particulars of this process.

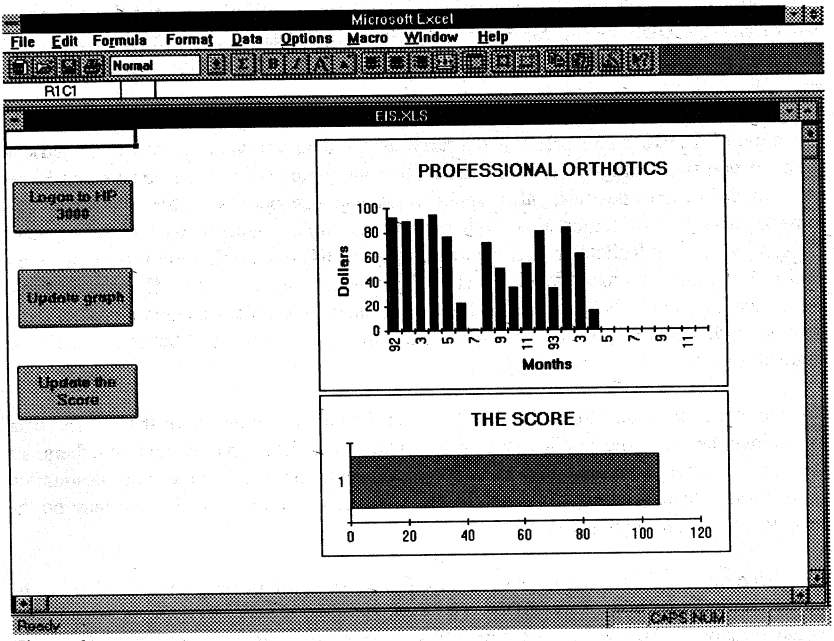


Figure 1

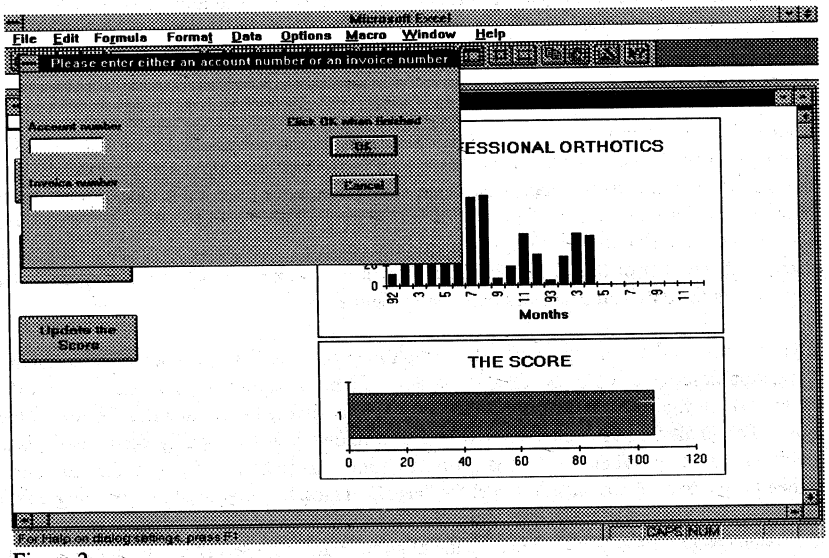


Figure 2

	1	2	3
55		UpdateGraph	
56		=SET.VALUE(R9C11,0)	*initialize dialog box values*
57		=SET.VALUE(R10C11,0)	
58	UpdateGraphBoxRe	=DIALOG.BOX(Inv_AcctSelectBox)	*activate dialog box*
59		=IF(UpdateGraphBoxResult=FALSE)	*abort if "cancel" is clicked*
60		=GOTO(StopUpdateGraph)	
61		=END.IF()	
62	AcctNumber	="01"&R[-53]C[9]	
63	InvNumber	="01"&R[-53]C[9]&"00"	
64	SysChan2	=INITIATE("R1WIN","SYSTEM")	*start a "system" DDE conversation*
65	RCLChan2	=INITIATE("R1WIN","RCL")	
66		=POKE(RCLChan2,"V5",AcctNumber)	
67		=POKE(RCLChan2,"V6",InvNumber)	
68		=EXECUTE(RCLChan2,"[invoke dde_two.rcl	
69		=WAIT(NOW()+"00:00:03")	
70		=REQUEST(SysChan2,"AppStatusValue")	
71		=SET.NAME("AppStat2",R[-1]C)	*check status of Reflection*
72		=WHILE(AppStat2<>0)	*loop while Reflection –*
73		=MESSAGE(TRUE,"Please wait while proce	*status <> "ready" and send –*
74		=REQUEST(SysChan2,"AppStatusValue")	*"Please wait" message to screen*
75		=SET.NAME("AppStat2",R[-1]C)	
76		=NEXT()	
77		=MESSAGE(FALSE)	
78		=TERMINATE(SysChan2)	
79		=TERMINATE(RCLChan2)	
80	StopUpdateGraph	=HALT()	

Figure 3

	5	6	7	8	9	10	11	12
1								
2	*item*	*x pos*	*y pos*	*width*	*height*	*item text*	*initial value/result*	*names*
3								
4								
5	Inv_AcctSelectBox							
6	5	10	6	575	202	Please enter either an account number or an invoice number		
7	5	10	51			Account number		
8	5	10	110			Invoice number		
9	6	10	69	100			0	
10	6	10	128	100			0	
11	1	405	70	88		OK		
12	5	349	50			Click OK when finished		
13	2	405	111	88		Cancel		

Figure 4

The macro in figure 3 may be intimidating to those who haven't used macros. I will step through it, explaining what is happening as I go. The image you see in figure 3 is a "display formulas" view, the actual values contained in the cells is shown in figure 5. Column one contains the range names that correspond to the cells immediately to their right. Column two contains the code of the macro. In this case the macro UpdateGraph is a range of cells from R55C2 to R80C2. The third column contains comment to help me remember what the code does.

	1	2	3	4	5
53	*names*	*formulas*	*comments*		
54					
55		UpdateGraph			
56		TRUE	*initialize dialog box values*		
57		TRUE			
58	UpdateGr	FALSE	*activate dialog box*		
59		TRUE	*abort if "cancel" is clicked*		
60		TRUE			
61		FALSE			
62	AcctNumb	01			
63	InvNumbe	01I4805500			
64	SysChan2	10	*start a "system" DDE conversation*		
65	RCLChan2	11			
66		TRUE			
67		TRUE			
68		TRUE			
69		TRUE			
70		3			
71		TRUE	*check status of Reflection*		
72		TRUE	*loop while Reflection –*		
73		TRUE	*status <> "ready" and send –*		
74		0	*"Please wait" message to screen*		
75		TRUE			
76		TRUE			
77		TRUE			
78		TRUE			
79		TRUE			
80	StopUpda	TRUE			

Figure 5

The formulas in R56C2:R57C2 reset the values in the dialog box to 0, if they weren't reset they would contain the last value entered into them. The values of the dialog boxes are used later in the macro, initializing them will prevent problems and confusion.

The formula in R58C2 activates the dialog box. The macro pauses here until the user finishes using the dialog box. The cells R59C2:R61C2 check to see whether the cancel box was clicked in the dialog box. Clicking cancel sets the value of the cell R58C2 to FALSE, clicking OK or hitting the enter key sets the value of the cell to TRUE. This value is stored in the range name UpdateGraphBoxResult. The three cells in rows 59 through 61 are a conditional that checks the value of UpdateGraphBoxResult. If the value is false the macro jumps to the named range StopUpdateGraph and encounters the =HALT() statement, =HALT() stops the macro execution. If the value of UpdateGraphBoxResult is true the cells in rows 60 and 61 are ignored.

In rows 62 and 63 I retrieve the values inputted from the dialog box and add some prefix and suffix information to make the values correspond to the information in our Image databases. The concatenated information is stored in the range names AcctNumber and

InvNumber. I usually think of the range names as variables. They act like variables when used in Excel macros.

Row 64 begins a DDE conversation with Reflection using *system* as a topic. The value that is returned is stored in the range name/variable SysChan2. The actual value as shown in figure 5 is the DDE channel number. That channel number is unique and can later be used to request information about Reflection's status.

Row 65 begins a DDE conversation with Reflection using RCL as the topic. The channel number for this conversation is stored in the variable RCLChan2. RCL is a Reflection Command Language DDE topic that allows us to transmit data to Reflection command language variables.

In rows 66 and 67 I use the RCL DDE channel to transmit (POKE) data (the variables AcctNumber and InvNumber) into Reflection Command language variables (V5 and V6). If all has gone well, I now have the information that was entered into the dialog box stored as variables in Reflection.

Row 68 uses the same RCL DDE channel to run a reflection command file, DDE_TWO.RCL in this case.

I know that the command file I have executed will take between 20 seconds and two minutes to execute, depending on how busy the HP3000 is. I could just pause macro execution for two minutes but that would be a waste of time in those cases where it only took 20 seconds. The next nine lines of code check whether Reflection is executing a command file, and displays a message while it waits for the file to complete execution. In row 69 I pause 3 seconds to give the command file a chance to get going and notify Reflection that it is executing.

Row 70 uses the system DDE topic to determine what Reflection is doing. Reflection keeps track of its status with internal variables we can access from Excel. When Reflection is executing a command file it sets its variable AppStatusValue to 3, when it is idle it has an AppStatusValue of 0.

Row 71 sets the variable/range name "AppStat2" to the value of the cell directly above it (R[-1]C). In row 72 I start a loop routine that continues while the value of AppStat2 is not equal to 0.

Row 73 displays a message in the bottom left corner of the screen while the loop continues.

Row 74 and 75 refresh the value of AppStat2 to the current value so that the loop will stop at the correct time. The loop doesn't include the cell in row 71 that first checked the AppStatusValue, if I didn't refresh the value of AppStat2 it would never stop looping.

Row 76 terminates the while loop I began in row 72. The rest of the macro can now execute.

Row 77 removes the message from the bottom of the screen asking us to please wait.

Rows 78 and 79 close the DDE channels we opened. It is important not to leave DDE channels open, they use system resources and there is a finite number of them. Probably the best reason is that you might mistakenly reference a channel you had used in another part of the program. Besides it's just sloppy programming to open things and not close them.

Row 80 terminates macro execution.

The UpdateGraph macro assumes that the user logged onto the HP3000 in Reflection with the appropriate files loaded. I could have added a routine that ran Reflection, logged on and loaded the proper files but security was a problem. I didn't want someone to click a button and get access without going through the proper security. Clicking the Logon to HP3000 button executes the LogonToHP3K macro(figure 6). This macro loads a dialog box to accept password information, then logs on to the 3000 and loads the proper files. Limitations of space preclude an in-depth explanation of that macro but it is very similar to the UpdateGraph macro.

Figure 7, the UpdateScore macro doesn't use a dialog box. This macro is activated by clicking the Update the Score button. It executes a Reflection command file that refreshes the small graph called The Score.

	A	B	C
1			
2	*names*	*formulas*	*comments*
3		LogonToHP3K	
4		=SET.VALUE(K19,0)	*initialize dialog box values*
5		=SET.VALUE(K20,0)	
6		=SET.VALUE(K21,0)	
7		=SET.VALUE(K22,0)	
8	BoxResult	=DIALOG.BOX(LogonInfoBox)	*activate the logon dialog box*
9		=IF(BoxResult=FALSE)	*abort if "cancel" is clicked*
10		=GOTO(EndMacro)	
11		=END.IF()	
12	SysChan	=INITIATE("R1WIN","SYSTEM")	*start a "system" DDE conversation*
13	AppStatusValue	=REQUEST(SysChan,"AppStatusValue")	*test the status of Reflection*
14	RCLChan1	=INITIATE("R1WIN","RCL")	*start a "command" link*
15		=POKE(RCLChan1,"V2",UserName)	*send Reflection the values --*
16		=POKE(RCLChan1,"V3",UserPassword)	*collected with the Logon --*
17		=POKE(RCLChan1,"V4",AccountPassword)	*dialog box*
18		=POKE(RCLChan1,"V5",GroupPassword)	
19		=EXECUTE(RCLChan1,"[invoke logon1.rcl]")	*log on to HP3000*
20		=WAIT(NOW()+"00:00:03")	
21		=REQUEST(SysChan,"AppStatusValue")	
22		=SET.NAME("AppStatus",B21)	*check status of Reflection*
23		=WHILE(AppStatus<>0)	*loop while Reflection --*
24		=MESSAGE(TRUE,"Please wait while initial processing takes place")	*status <> "ready" and send --*
25		=REQUEST(SysChan,"AppStatusValue")	*"Please wait" message to screen*
26		=SET.NAME("AppStatus",B25)	
27		=NEXT()	
28		=MESSAGE(FALSE)	
29		=TERMINATE(SysChan)	*close the DDE conversations*
30		=TERMINATE(RCLChan1)	
31	EndMacro	=HALT()	*terminate the macro*

Figure 6

	A	B	C
104	*names*	*formulas*	*comments*
105			
106		UpdateScore	
107	SysChan3	=INITIATE("R1WIN","SYSTEM")	*start a "system" DDE conversation*
108	RCLChan3	=INITIATE("R1WIN","RCL")	
109		=EXECUTE(RCLChan3,"[invoke calcscor.rcl]")	*execute the RCL command file – calcscor.rcl*
110		=WAIT(NOW()+"00:00:03")	
111		=REQUEST(SysChan3,"AppStatusValue")	*check the status of Reflection*
112		=SET.NAME("AppStat3",B111)	
113		=WHILE(AppStat3<>0)	*loop while status is busy*
114		=MESSAGE(TRUE,"Please wait while processing takes place")	
115		=REQUEST(SysChan3,"AppStatusValue")	
116		=SET.NAME("AppStat3",B115)	
117		=NEXT()	
118		=MESSAGE(FALSE)	
119		=TERMINATE(SysChan3)	
120		=TERMINATE(RCLChan3)	
121		=HALT()	

Figure 7

The Reflection component

Reflection 1 for Windows, version 4, provides the link between the PC and the HP3000. Reflection can provide a PC/3000 link in two ways. The first method is file transfer, I could write a command file that would send information back and forth. This might work well for some applications but it is unnecessarily complicated for this application. The second method of transfer, the ability to store the contents of specific screen areas in variables, was just what I needed for this application.

Figure 8 shows the Reflection command file DDE_TWO.RCL. This file uses the information that Excel transmitted to the Reflection variables V5 and V6. Section 1 loads values into QueryCalc and does the calculations. The third line of the file tells QueryCalc to make the cell A1 the active cell, "/J A1^M" tells QueryCalc to jump to A1. The ^M is the RCL equivalent of hitting the return key. The fifth line -- Transmit "'" & V5 & "^M", enters an apostrophe followed by the value of the variable V5, and then a return. The apostrophe tells QueryCalc to accept the value following as text. The next few lines move down one row and enter the value of the second variable. The line Transmit "!!A1:D15^M", tell QueryCalc to calculate the block of cells in A1:D15. The double exclamation point tells QueryCalc to calculate query questions as well as more typical spreadsheet calculations.

Section 2 initiates a DDE conversation with the Excel spreadsheet EIS.XLS and stores the value of the DDE channel in the variable V0. The nine lines after the DDE-INITIATE line store the values of screen areas in variables. The statement LET V1 = SCREEN(5,16,5,27) means: assign the value contained in the block of screen beginning at row 5 and column 16, and ending at row 5 and column 27 to the Reflection variable V1.

The following nine lines "POKE" the information into Excel. The statement DDE-POKE V0 "R1C30" "$1", means: take the RCL variable V1(V1 translates to $1) and place it into the Excel spreadsheet EIS.XLS at row 1, column 30. The V0 directs the DDE-POKE to

use the DDE channel we opened with the DDE-INITIATE command. Because it is possible to have several DDE conversations active at the same time, Reflection insists that you specify the channel you want to poke the information through.

Reflection only allows variables V0 through V9 to be poked to other applications. This means we need to assign a group of variables, poke them through to Excel, then reassign them in small groups until all the information is moved. This happens in the background and goes fairly quickly, how quickly depends on the CPU, memory, disc speed, etc., in your PC.

The line LET V9 = MID(V8,1, FIND("", V8)-1) is a special case. The variable V8 is storing the name of a customer. In the past we coded our customer names with special characters surrounded by quotes. These quotes cause problems when I try to perform the DDE-POKE. They are interpreted as the end of a string rather than a part of one. I could replace them with different characters that didn't cause problems but in this case it is simpler to just remove them. The statement above takes the value in variable V8 and stores everything to the left of a quote mark in the variable V9. I realize that this isn't a terribly elegant solution to this problem, but the fact that it works at all endears it to me.

The second to last line DDE-TERMINATE V0 closes the DDE channel and stops the conversation.

Each of the Excel macros has a corresponding Reflection command file that is similar to the one in figure 8.

```
SET DDE-TIMEOUT 15
Transmit "/J A1^M"
Hold For "^[J^Q"
Transmit "'" & V5 & "^M"
Hold For "^[J^Q"
Transmit "/J A2^M"
Hold For "^[J^Q"
Transmit "'" & V6 & "^M"
Hold For "^[J^Q"
Transmit "!!A1:D15^M"
Hold For "^[J^Q"
WAIT 0:0:3
DDE-INITIATE "EXCEL" "EIS.XLS" V0
LET V1 = SCREEN(5,16,5,27)
LET V2 = SCREEN(6,16,6,27)
LET V3 = SCREEN(7,16,7,27)
LET V4 = SCREEN(8,16,8,27)
LET V5 = SCREEN(9,16,9,27)
LET V6 = SCREEN(10,16,10,27)
LET V7 = SCREEN(11,16,11,27)
LET V8 = SCREEN(12,16,12,27)
LET V9 = SCREEN(13,16,13,27)
DDE-POKE V0 "R1C30" "$1"
DDE-POKE V0 "R2C30" "$2"
DDE-POKE V0 "R3C30" "$3"
DDE-POKE V0 "R4C30" "$4"
DDE-POKE V0 "R5C30" "$5"
DDE-POKE V0 "R6C30" "$6"
DDE-POKE V0 "R7C30" "$7"
DDE-POKE V0 "R8C30" "$8"
DDE-POKE V0 "R9C30" "$9"
LET V1 = SCREEN(14,16,14,27)
. . .
```

(The pattern repeats as necessary in this space)

```
. . .
LET V7 = SCREEN(2,27,2,33)
LET V8 = SCREEN(3,39,3,67)
LET V9 = MID(V8,1, FIND('"', V8)-1)
DDE-POKE V0 "R19C30" "$1"
DDE-POKE V0 "R20C30" "$2"
DDE-POKE V0 "R21C30" "$3"
DDE-POKE V0 "R22C30" "$4"
DDE-POKE V0 "R23C30" "$5"
DDE-POKE V0 "R24C30" "$6"
DDE-POKE V0 "R2C28" "$7"
DDE-POKE V0 "R3C28" "$9"
DDE-TERMINATE V0
;END OF PROGRAM
```

Figure 8

The HP3000 side

I use QueryCalc from AICS Research to perform the queries in my application. QueryCalc is a report writer/3-D spreadsheet program. In QueryCalc any cell can contain text, numbers, or a query question. QueryCalc is highly optimized for speed. While speed is always important, in a client-server application it is crucial. I also like the fact that QueryCalc is so robust, powerful and flexible. The fact that QueryCalc looks like a 3-D spreadsheet lends itself to my application. I can input values into cells and have a query question in another cell reference those values as criterion. The spreadsheet format also allows me to know the exact location of the information I need to load into Reflection variables.

Figure 9 shows the equations from the QueryCalc worksheet. The cells in Aa1:Ad15 are used in the example I discussed above. The database I need to Query has an account number (cust-key) field, but not an invoice number field. The cell Aa1 will contain an account number if one was entered in the Excel dialog box, otherwise it will contain a zero. The cell Aa2 will contain an invoice number if one was entered in the dialog box, again it will contain a zero otherwise. The query question in cell Aa3 provides an account number for the invoice-number in cell Aa2.

```
Equations for Page A
____________________


   (Aa3): @USING ORDERS.ORDR-DETL, VAL OF CUST-KEY WHEN ORDER-
NUMBER=[$A2]
   (Aa4): @USING ORDERS.CUST-DETL, STORE IN A CUST-KEY WHEN CUST-
KEY=[$A1],[$A3]

   (Ab1): @USING ORDERS.CUST-DETL, VAL OF CUST-KEY WHEN CUST-KEY=!A
   (Ab3): @USING SALES.SALES-DETL, FIND WHEN CUST-KEY=!A AND FISCAL-
PERIOD>=0119
9201
   (Ab4): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199201
   (Ab5): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199202
   (Ab6): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199203
   (Ab7): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199204
   (Ab8): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199205
   (Ab9): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199206
  (Ab10): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199207
  (Ab11): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199208
```

```
  (Ab12): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199209
  (Ab13): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199210
  (Ab14): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199211
  (Ab15): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199212
   (Ac4): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199301
   (Ac5): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199302
   (Ac6): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199303
   (Ac7): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199304
   (Ac8): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199305
   (Ac9): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199306
  (Ac10): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199307
  (Ac11): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199308
  (Ac12): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199309
  (Ac13): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199310
  (Ac14): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199311
  (Ac15): @REREADING, SUM OF DOLLARS/10000 WHEN FISCAL-PERIOD=01199312
   (Ad1): $SUB$(B1,3,8)
   (Ad2): @USING ORDERS.CUST-DETL, VAL OF NAME WHEN CUST-KEY=[$B1]
   (Af1): $DAT$2(SYSDATE)
   (Af2): @USING ORDERS.ORDR-DETL, STORE IN A ORDER-NUMBER WHEN ORDER-
DATE=[$F1]
 AND STATUS<>CN
   (Af3): @USING ORDERS.ORDR-DETL, STORE IN B ORDER-NUMBER WHEN ORDER-
NUMBER=!A
AND DOCUMENT-TYPE<>Z
   (Af4): @USING ORDERS.ITEM-DETL, SUM OF QTY-ORDERED/10000 * UNIT-
SELL-PRICE/10
000 WHEN ORDER-NUMBER=!B AND LINE-
          TYPE<>E@
   (Af5): @USING ORDERS.ORDR-DETL, STORE IN C ORDER-NUMBER WHEN ORDER-
NUMBER=!A
 AND DOCUMENT-TYPE=Z
   (Af6): @USING ORDERS.ITEM-DETL, SUM OF QTY-ORDERED/10000 * UNIT-
SELL-PRICE/10
000 WHEN ORDER-NUMBER=!C
   (Af7): F4-F6
```

Figure 9

The cell Aa4 creates a "search set" of the account numbers in cell Aa1 or Aa3. The idea of a search set may be unfamiliar to you. The basic idea is create a list of values that be used in subsequent queries. Because zero is not a valid account number (either Aa1 or Aa3 will be a zero and the other cell will contain an account number) our search set contains one valid account number. The search set is assigned the name A, valid search set names are A through Z. The cell Ab1 returns the value of the account number in search set A, we'll use this a little later on. The cell Ab3 uses the account number stored in search set A to query the database that contains my historical sales data. This cell gathers the records that have the appropriate account number and are from January of 1992 or later.

In the block of cells Ab4:Ac15 I use the QueryCalc @rereading function to find the total dollar sales in each fiscal period. The @rereading function is something like the Query subset function but is more flexible. The @rereading function does not disturb the original group of records selected and can be reissued many time with different qualifying criteria. Reading through a list of records is much faster than asking a query question repeatedly. Search sets and the @rereading function are two of many features that make QueryCalc well suited for an application such as this one.

The cell Ad1 parses out the account number in cell Ab1, removing the 01 prefix it has in the database. The value of this cell is passed back to Excel.

Cell Ad2 returns the value of the account name for the account in question. This information is sent back to Excel for display in the graph.

The group of cells Af1:Af7 calculate the up to the minute sales for the day. The first cell in the group returns the current date in the proper format. The second cell uses that date to create a search set of all the invoice numbers for the day (this is search set A).

Cell Af3 further refines the search set to filter out credit invoices. Cell Af4 returns the dollar value of sales represented by search set B. Cell Af5 creates a new search set of invoice numbers for credit invoices. Cell Af6 returns the dollar value for the credit invoices. Cell Af7 subtracts the credit invoices from the sales invoices to give the net sales for the day. It may appear that separating the sales and credit invoices, then netting them is redundant. It is necessary in this case because the values in the database for a credit invoice are not negative numbers even though the credit invoice is essentially a negative sale.

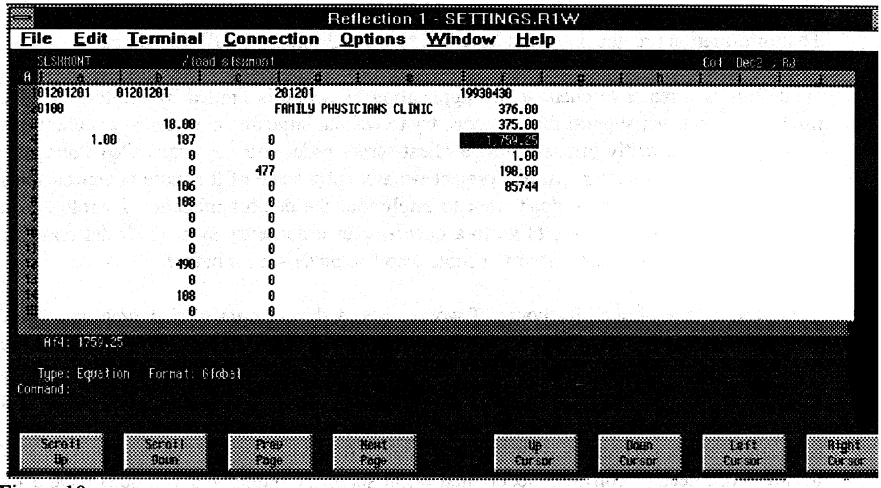


Figure 10

The final value in Af7 is transmitted back to Excel where it is displayed in the graph titled THESCORE.

While neither of these examples used it, QueryCalc has a strong macro language that can be helpful. In an application that required a lot of manipulations of the spreadsheet a macro within QueryCalc would be much faster and more reliable than issuing the individual commands from the Reflection command file.

QueryCalc also has the ability to print to MPE flat files. If all the information needed couldn't fit on the screen it could be printed to a file then transferred to the PC with the Reflection command file. I avoided this because it added an unnecessary layer of complexity. Transferring the value of variables with DDE is much easier than transferring files, bringing the file into Excel, parsing the ASCII data, and then manipulating the

information within Excel. The DDE data transfer is also much faster than a file transfer. File transfer data exchange can work but it adds a whole list of new things that can go wrong with your application. Use file transfer if you have to but avoid it if you can.

AICS Research has just added a new feature to QueryCalc that could simplify the process even further. They have developed high quality PostScript graphics for QueryCalc. Rather than bringing the information into Excel to generate the graphics I could just download the PostScript files, then use something like Adobe's Acrobat to convert them into screen displayable images.

Closing thoughts --

The application I have discussed is meant to be a demonstration of what client-server can do. Perhaps the thing that surprised me the most was how much people liked having this application available to them. Everyone that saw it wanted something like it for themselves. Everyone loved the fact that it could provide them information on demand. That information previously would have had to come through the DP department.

While this is a real live client-server application its scope is limited by the tools I have used. I would never suggest that someone try to write a large and complex application this way. We are currently implementing a client-server order-entry system called Point.man from Spectrum Associates. In this project we are using some of the more advanced tools such as Gupta. I certainly don't want to imply that the demo application I wrote had a major impact on our decision to go to a client-server order entry system. It did however show the potential of client-server to people who had never seen it before.

This project took about fifty hours of work. About thirty hours of that time was spent trying to find the correct syntax for the DDE commands. There were cases where I had to resort to trial and error to find the right combinations. I hope that the examples I have provided will allow people to put together a demonstration client-server application in a very short period of time. Using a product like Forest & Trees is certainly a better long-term solution to providing desktop access to data. It is designed for that very purpose. If however you need to convince people that client-server is viable, and you don't have a lot of money to throw at the project, try this DDE method.

#7030

"What's So Hard About Software on CD?": A CD-ROM Information Publishing Primer

Katherine J. Armstrong

Hewlett-Packard Company
690 E. Middlefield Rd., M/S 30LR
Mountain View, CA 94043
(415) 960-5844

1. Introduction

The purpose of this paper is to provide a high-level understanding of CD-ROM technology and the issues associated with its use for software and documentation distribution. This paper provides a starting point for readers who want to learn how to make profitable use of this technology in their business.

Section 2 provides a brief technical overview of the CD-ROM medium, followed, in sections 3 and 4, by a discussion of the pros and cons involved with using this medium for information distribution. The specific steps involved in transferring information onto CD-ROM, and some suggestions for how to get started, will be outlined in Section 5. Section 6 will focus on issues specific to distributing documentation (text and graphics) on CD-ROM. Finally, we will conclude with some "words of wisdom" based on our experiences with using CD-ROM for software and documentation distribution at Hewlett-Packard.

2. CD-ROM: History, Terms, and Standards

2.1 A Bit of History

During the early 1980s, compact discs made a splash in the consumer music market, primarily because of their capability for high-fidelity reproduction, their durability, and the attractive economics of high-volume CD manufacturing. CDs quickly gained acceptance as the preferred distribution medium for music by the end of the 1980s. This early success inspired CD manufacturers to look for other applications of this technology; at the same time, the computer industry was beginning to struggle with storage capacity problems resulting from the newly developing interest in multi-media. By the mid-1980's, it had become apparent that CD technology had considerable potential for computer industry applications as well.

2.2 CD Standards

As the use of CD technology in many industries has continued to grow, researchers have developed standards that apply to both the hardware (physical) and data format (logical). Adherence to industry-approved standards, as opposed to developing proprietary schemes, has facilitated cross-platform information interchange to a degree not often seen in the computer industry.

Physical standards define the protocols for hardware interaction. These low-level definitions enable the CD drive to read data from the CD media. The current physical CD standards, most of which were developed by Philips and Sony, include:

- **Red Book:** Compact Disc - Digital Audio (CD-DA). This standard (IEC 908) is used by the music industry to publish your favorite titles at the "record" store. Discs formatted to this standard are usually stamped with "Compact Disc Digital Audio".

- **Yellow Book:** Compact Disc - Read Only Memory (CD-ROM). The Yellow Book standard for CD-ROM (ISO/IEC 10149:1989) is the most commonly used CD standard for distributing information in the computer industry. Titles that conform to this standard will usually be stamped with "Compact Disc Data Storage". An extension to the Yellow Book standard, "CD-ROM/XA," provides for compressed audio and video/picture data in addition to the standard computer data stored on CD-ROM.

- **Orange Book:** Recordable Compact Disc. The entrants in this hodge-podge standard include CD-MO (Magneto-Optical), CD-WO (Write Once), CD-R (Recordable), and multi-session "Hybrid Discs". The Orange Book "standard" continues to evolve, and is thus a frequent subject of confusion. Its areas of application include systems that produce "one-offs", used primarily by developers who need to "burn" a CD for testing prior to shipping to a vendor for mastering and replication. Kodak's new Photo-CD will also adhere to this standard.

- **Green Book:** Compact Disc - Interactive (CD-I). This standard is part of the emerging multi-media industry. CD-I is a comprehensive software and hardware system specification for storing text, graphics, and audio. The CD-I technology was designed for home use: CD-I players can be plugged into your television and stereo.

- **Blue Book:** This is the Laser Disc standard for full motion video and audio.

The above standards cover a broad range of compact disc technologies; in this paper we will focus exclusively on CD-ROM. (A good technical comparison of these standards is available in [4].)

The physical layout of the disc is only part of the CD-ROM story. A protocol is also needed for how the operating system is to interpret the information stored on a CD-ROM. For software and documentation publishing, we have the ISO-9660 file system standard and the emerging RockRidge format, which extends ISO-9660 for UNIX systems.

ISO-9660. This is one of the most widely implemented cross-platform standards in the industry. ISO-9660 emerged from the High Sierra Group recommendations and was formalized into the standard with a few modifications. The High Sierra Group, and the *High Sierra format* that emerged from that group, were named for the High Sierra Hotel at Lake Tahoe, site of the first meeting held to formulate the standard. Although there are some minor differences between the original High Sierra format and the current ISO-9660 standard, you may still hear the two names used interchangeably.

ISO-9660 (or High Sierra), standardized in 1988, was developed to provide system interoperability for CD-ROM. The ISO-9660 file system looks much like a DOS file system, however there are different levels of implementation and interchange. At the most limiting level,

filenames are at most eight characters in length, followed by a period and a three character extension. No special characters are allowed and directories can only be eight levels deep. It is important to note that different operating systems choose different levels of implementation of ISO-9660, and thus have different file system limitations. Not all ISO-9660 disks are interchangeable across all systems that support the standard. You must check the implementation level for compatibility.

RockRidge. UNIX file systems are more complex than early PC file systems. Filename conventions are less restrictive, and files have attributes for ownership, permissions and special files. The RockRidge extension to ISO-9660 was developed to provide these much needed features that were missing from the ISO-9660 standard. RockRidge layers POSIX file system semantics on top of ISO-9660, allowing UNIX-like file systems to live on a CD-ROM and be read just like any other mounted disk on the system. One of the key advantages RockRidge provides is the ability to execute UNIX software directly from the CD-ROM, without first installing the software onto your hard disk. RockRidge is currently working its way through the standards process. However, many vendors have already implemented RockRidge in their operating systems.

2.3 Multi-media and Other Trends

Current trends in CD-ROM technology include recordable and multi-session CD-ROM -- where data can be appended to a previously recorded disc -- and multi-media applications that include sound, animation, and full-motion video. New standards and technologies being developed to support these types of applications include CD-I, CD/XA, and Photo-CD. These technologies are still in their infancy, however, and the commercial availability of authoring tools to facilitate multi-media development is slow. The result is that multi-media titles continue to be extremely expensive and time-consuming to make, generally out of reach for all but the most specialized and sophisticated of applications.

3. The Case for CD-ROM Distribution

In browsing through any of the major PC or workstation journals, it becomes immediately clear that CD-ROM is a "hot" topic. But why all the interest?

The advantages of using CD-ROM can be categorized, with some overlap, as either business advantages or customer advantages. Historically, business reasons, such as cost savings, have been the initial triggers for adopting CD-ROM for information distribution; however, recently there has been a shift towards the customer perspective.

3.1 Business Advantages

Among the business reasons for adopting CD-ROM for software and documentation distribution are:

- **Cost-savings.** In larger volumes, CD-ROM materials are significantly less expensive than other distribution media. After paying the initial set-up charges (see Section 5.1 for more on CD-ROM manufacturing), CD-ROMs can be duplicated for $1-2 each, with packaging and shipped adding another $2-5, depending on complexity. Contrast this with magnetic tape,

which must be recorded serially, yielding no significant savings for volume replication, or paper, which can be duplicated cheaply, but for which the packaging and shipping charges quickly mount up. As an example, one of our HP divisions estimated their manufacturing cost for paper manuals, not including shipping, at over $150,000. This division recently made the switch to CD-ROM, and is now spending under $30,000, *including packaging and shipping*.

- **Time-savings.** The CD-ROM replication process is a simple "widget-stamping" procedure. As such, it is much faster to produce CD-ROM media in volume than to duplicate tape media or paper. Shortened manufacturing times can translate into shorter lead time requirements for application and learning products developers and decreased time-to-market for new products.

- **Capacity.** A single CD-ROM can store almost 700 MBytes of information, compared with traditional distribution media, such as floppy disks (1.44 MBytes) or 9-track magnetic tape (100 MBytes). To put this in real terms, a full CD-ROM can replace approximately 1000 pounds worth of paper manuals, saving 40 feet worth of shelf space. As shipping costs continue to increase, this "compact" property of CD-ROM can result in significant additional savings for businesses through decreased packaging and shipping costs.

- **Multi-platform distribution capabilities.** CD-ROM is unique among distribution media in the existence and wide-spread adoption of cross-platform file system standards for this medium. A CD-ROM formatted according to these standards can be mounted and read across multiple hardware and operating system platforms.

3.2 Customer Advantages

Benefits for end-users of CD-ROM-based software products include:

- **More efficient software installation and update.** Because CD-ROM is a random-access medium containing a file system image (versus tape, which must be read sequentially), end-users can partially install or update large software products directly from the medium, avoiding the unnecessary read time associated with sequential media. PC users used to repeatedly changing floppies during installation will find that this annoyance goes away when installing from CD-ROM. Another advantage of CD-ROM for software installation is that the disc can be mounted as a network file system and accessed remotely.

- **Fast and convenient access to information.** Searching for information is a considerable source of lost time in an organization. Hewlett-Packard's Microwave Instruments Division estimates that, without electronic access to documentation, its service engineers lose approximately one-third of their day to searching for information. CD-ROM, along with the appropriate indexing and retrieval software, facilitates efficient electronic access to huge quantities of information. For individuals, this means increased productivity. Quality of work increases too, as individuals are more likely to take the time to gather the information they need to do their jobs if it is conveniently available at their desktops. For businesses, this can also translate into better service for your customers.

- **Bringing together information from diverse sources.** As we have found at Hewlett-Packard, customers like the fact that with HP LaserROM CD-ROMs they have at their fingertips not only all of the manuals for the products they have purchased, but also manuals

for many other products, technical notes, product catalogs, and much more. With CD-ROM, customers have cost-effective and convenient access to many more sources for information than they have had with traditional media.

- **Space savings.** Hewlett-Packard's LaserROM/UX CD-ROM contains over 200 manuals. The paper versions of these same manuals would require shelf space covering an entire cubicle wall. For system administrators, support engineers, service technicians, and other personnel whose jobs require access to large amounts of technical documentation, this is a very important advantage.

- **Sharability.** Manuals can only be used by one person at a time. They can also be lost or borrowed, making them inaccessible when you need them. To counteract this, organizations often buy multiple copies of important manuals. In contrast, by purchasing a single CD-ROM service, everyone in the organization may have access all the time to on-line information.

- **Durability.** Compared to paper for documentation, or to magnetic media for software, CD-ROMs are relatively indestructible. They are not easily damaged by scratches, smudges, being dropped, or other handling artifacts. They can withstand fairly extreme variations in temperature without degrading the quality of their data, and their lifetime expectancy is considerably longer than that of magnetic tape [3].

- **Preserving natural resources.** The use of CD-ROM in place of paper to distribute documentation saves trees. Hewlett-Packard, along with other companies dedicated to preserving the environment, is also initiating recycling programs for both the media itself and for the packaging, including plastic jewel cases.

4. Obstacles to Adopting CD-ROM

While we have looked at the advantages CD-ROM may provide for companies and their end-users, there are also reasons why adopting CD-ROM as a distribution medium may not be right for your company. It is important to understand that there are hurdles associated with the adoption of a CD-ROM program, which both your company and your customers must be willing to negotiate and overcome. In this section, we will discuss some of the negatives you may encounter from your management and/or your customers, and suggest alternative perspectives to help you counter these objections.

4.1 Business Considerations

Start-Up Cost. Business considerations regarding whether to implement a CD-ROM program often come down to a simple question of economics. Will the long-term savings associated with using CD-ROM in place of other distribution media be worth the initial set-up costs? To find the answer to this question, you must first understand what *are* the initial set-up costs. This topic will be discussed in detail in Section 5, but the point that we will make here is that you must be able to convince your management (and yourself!) that the long-term cost savings for your company will be worth the start-up investment required.

Paradigm Shift. If you intend to distribute documentation on CD-ROM, you must keep in mind the costs associated with the paradigm shift from paper manuals to on-line technical

documentation. I call this a "paradigm shift" and not simply a migration because, in order to make the shift without negatively impacting your customers, your technical writers and editors must adopt a different way of thinking about information presentation. (This topic will be discussed in more detail in Section 6.) This paradigm shift may require a change in authoring tools, it often necessitates a (sometimes major) data conversion effort, and it may even lead to reorganization (or establishment) of an editorial staff. The key to acceptance of this paradigm shift within your organization lies in communication, education, and partnership between decision-makers, developers, and production staff.

4.2 Customer Considerations

Cost of Drive. The most obvious end-user obstacle you may face is the requirement that all of your customers have CD-ROM drives. As we have found at Hewlett-Packard, there is still a perception among consumers that CD drives are very expensive. In fact, low-end drives can easily be purchased for a few hundred dollars. When viewed in the context of the purchase price of an entire computer system -- even a PC system -- this incremental cost is very minimal. Further, the availability of CD-ROM products is growing rapidly, making the purchase of a CD-ROM drive more of a strategic investment. The installed base of CD-ROM drives more than doubling between 1991 and 1992, and the head of Microsoft's consumer division believes that within two years all PCs will be equipped with CD-ROM drives [5]. Hewlett-Packard and other companies have found that promotional programs giving away CD-ROM drives can be an effective way to increase CD-ROM penetration into the customer installed base. In the long-run, the cost savings HP has experienced using CD-ROM for information distribution has easily outweighed the expense associated with these promotional programs.

Customer Value Perception. A related, and perhaps more serious, obstacle is that customers do not see a compelling reason for them to purchase a CD-ROM drive. You may experience an attitude along the lines of: "I don't have a problem, so why do I need a solution?" The way to counteract this problem is communication. Customers must be made aware of the advantages to *them* of receiving their software and documentation on CD-ROM. They must not perceive the move to CD-ROM as simply a cost-cutting measure for you, the supplier. This is important to keep in mind. *There must be real and perceived benefit for the end-user, else attempts to initiate a CD-ROM program will not succeed.*

On-line versus Paper Documentation. If your target audience is highly technical, you may not encounter a lot of resistance to switching from paper documentation to on-line. However, some of the concerns you may hear from customers, and some suggestions for addressing these concerns, include:

- On-line documentation is not portable. I have to be sitting at my desk to get at the information I need.

  Response: The retrieval software you provide with your CD-ROM should have flexible print functionality.

- I can dog-ear and mark up my paper manuals, but CD-ROM is read-only.

  Response: The retrieval software should provide bookmark and annotation features.

- I'm used to finding my way around in a book. How do I find my way around a CD-ROM-full of information?

Response: The retrieval software should have a range of intuitive navigational aids to facilitate browsing, in addition to full-text search capabilities.

- This product can't possibly be worth this much if all the documentation fits on this little disc.

  Response: Introduction of your CD-ROM program should include broad-based marketing communications programs to make clear the benefits to end-users of on-line documentation. For future releases, you might also consider adding computer-based training software and other educational aids to demonstrate the added-value of CD-ROM over paper documentation.

5. Putting Information on CD-ROM

5.1 The Process

The process of putting information on a CD-ROM is outlined in Figure 1 below.

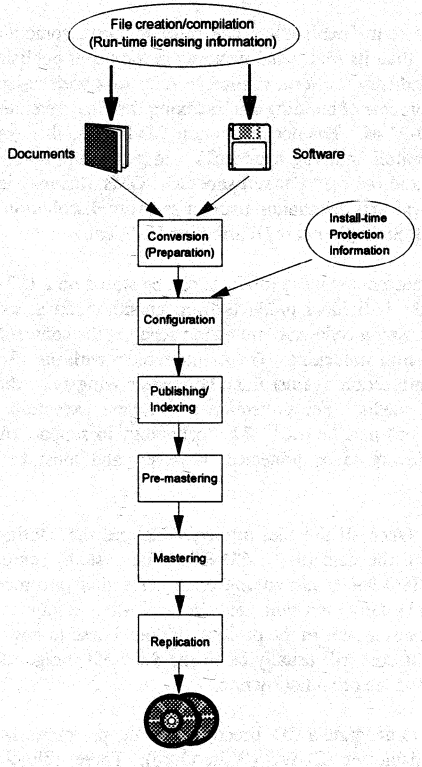


Figure 1. CD-ROM Production Process

The basic steps in CD-ROM production are:

1. **Conversion** -- Text and graphics must be captured in electronic form and converted into whatever format is required by your indexing and/or display programs. Any software that is to be placed on the CD-ROM must be prepared for the target platform in the same manner as with other distribution media.

2. **Configuration** -- Configuration involves gathering together the files to be placed on the CD-ROM and specifying how they are to be arranged on the disc to facilitate retrieval. For both software and documentation, configuration may include specifying a file system or database structure and building control files for indexing and publishing engines. For simple applications not requiring full-text indexing, or simple software product configurations, it may suffice to organize your files into an ordinary file system structure, as software products and files can be located on the CD-ROM without the need for indexing. For more complex configurations consisting of many software products and/or many MBytes of documentation, you may need to overlay some kind of database structure on top of the files to aid efficient retrieval. In this case, you will likely need to build control files for use by an indexing program as part of the configuration process.

3. **Publishing** -- The term "publishing" has come to mean something quite different in the computer industry than its traditional meaning in the paper publishing world. Further, even in the computer industry the term is used to refer to a wide range of activities. What we mean here is the process of building and indexing the final versions of all files that are to be placed on the CD-ROM. For documentation CD-ROMs, this may include indexing your data for full-text search, building hyperlinks, and generating other navigational aids such as tables of contents and indices. For software CD-ROMs, this may include indexing your files for efficient retrieval and installation using a software distribution utility such as the OSF's emerging standard, SDU (Software Distribution Utilities).

4. **Protection** -- Because many more products can be stored on a CD-ROM than on traditional media, securing the individual products from unauthorized access becomes a particularly important issue. Using a codeword protection scheme, the contents of your CD-ROM can be secured for install-time unlocking. The main problem with install-time security, however, is that it offers no protection against unauthorized copying once the data has been installed from the original media. For protection at software execution time, run-time licensing (password protection) may be used. The technology to support run-time licensing must be built into the software to be protected, however, and must be supported by the target operating system.

5. **Pre-mastering** -- Once all the files are available and the configuration set, the next step involves formatting the data into a CD-ROM file system, using a CD-ROM file system standard such as ISO-9660, and adding error correction information. This step is called "pre-mastering". Usually, you will generate a pre-master tape, which will be transferred onto a CD-ROM master later in the process. If you choose to have an outside vendor do the pre-mastering, your cost will usually be in the $200-500 range, depending on whether the vendor charges a flat fee or an hourly rate.

   It is also possible to generate a CD directly out of the pre-mastering step using technologies like CD-R (Recordable) or CD-WO (Write Once). These technologies allow "one-offs", as they are called, to be created in-house using special recording equipment costing as little as

$5,000. Many CD-ROM manufacturers will also generate one-offs, usually for $200-300 each. The blank media is more expensive than CD-ROM, and the manufacturing process is different, but the end result is completely compatible with CD-ROM, and can be read from any standard CD-ROM drive. These technologies can be a good alternative to CD-ROM for limited edition CDs, prototypes, and in-house archiving. (For more information on Recordable CD-ROM, see [7].)

6. **Mastering** -- A CD master is a glass disc that is encoded with your pre-mastered data through a special process performed at a mastering facility. A mold is created from the master disc, and this mold is used to press copies of your CD-ROM. Typical mastering costs are in the $900-1500 range.

7. **Replication** -- Bulk quantities of CD-ROMs are stamped out very cheaply once a mold is created. CD replication is typically performed by the same facility that built the master. Costs vary, depending on quantity, packaging, turn-around time, and other considerations, but $1-2 per disc is typical. (For technical details about the CD-ROM manufacturing process, see [8].)

5.2 How to Get Started

There are some general guidelines that will help you get started on the right track with your CD-ROM program. At Hewlett-Packard, a lot of experience has taught us these lessons. In sharing them here with you, we hope to make your transition to CD-ROM easier.

Get a quick education. Your program will get off the ground faster and with fewer problems if you start out with some understanding about the technology and process of building CD-ROMs. Reading this paper is a good place to start. Talking to people who have already gone through the process is another excellent way to gather information.

Be realistic about time and effort. Don't kid yourself that the problems others have experienced in getting their programs going won't apply to you. Know the potential pitfalls, and be prepared to address them. Conversion of legacy data, for example, is often a very expensive and time-consuming effort. Planning for it, as we have at Hewlett-Packard, can significantly bring down the cost and ensure that your program is launched on schedule. Administering your software licensing scheme is another area that often presents problems. Once again, look for help from people who are doing these things already.

Invest in prototyping. Whether you choose to build your own CD-ROM publishing and retrieval software, or you decide to take advantage of a third party's offering, take the time for a round of prototyping. Prototyping gives you the opportunity to make sure your design fits the needs of your customers, and helps you assess the feasibility of your ideas and get a handle on costs and potential difficulties. Many vendors will build a prototype for you, usually for a small fee.

You don't need to go it alone. Your introductory education should include a survey of the industry to find out who can help you with some or all of the implementation of your program. One of the easiest and fastest ways to do this survey is to attend conferences on CD-ROM, multi-media, or on-line publishing. Roam the exhibits floor to find out who is doing what. Attend talks -- not only to hear the speakers, but also to learn from the experience of other attendees who have already implemented CD-ROM programs.

5.3 Considerations for In-House Production

Many companies today, particularly those without in-house expertise in CD-ROM or on-line technologies, are electing to work with one or more specialists (service bureaus, tools providers, consultants, etc.) during the initial implementation of their CD-ROM program. Specialists can fill in knowledge gaps in your organization and get your program off the ground more quickly. For organizations that choose to go it alone, however, we have highlighted below some of the issues you will want to consider.

Hardware requirements. Unless you will be mass-producing extremely large quantities of CD-ROMs, it generally does *not* make sense for a company to invest in its own CD-ROM mastering or replication equipment. What may make sense, however, is to invest in a CD Recordable system for small volume custom CDs and prototypes. For larger volume applications, it is usually cheapest to contract with a third party for mastering and replication services. In general, you will need to assess your goals and long-term plans to determine what hardware purchases make sense for you.

Publishing and delivery software. There are several pieces of software you may need to develop or purchase for your CD-ROM program. You will need pre-mastering software to build CD-ROM format (e.g. ISO-9660) file system images from your files. You will need indexing software to prepare your data for search and retrieval. You will need publishing software to build hyperlinks and other navigational aids and prepare your text and graphics for on-line display. Options, ranging from least expensive and least flexible to most, include pre-packaged software, toolkits, custom third party development, and in-house development. The best solution is usually the cheapest and least customized solution that still meets your company's and end-users' needs.

Conversion costs. Format conversion is often the most expensive and time-consuming element in initiating a CD-ROM program for documentation distribution. Some information to be placed on the CD may be available only in hardcopy, and must first be captured electronically -- either by scanning and OCR (Optical Character Recognition), or through manual retyping. Text that is not in the format required by your publishing and indexing software must be converted. (See Section 6.3 for a discussion on options for conversion.) Similarly, graphic images may need to be scanned or regenerated and converted to a format supported by your on-line presentation software.

Information/configuration management. Managing the content of a CD-ROM is an important task that is often overlooked in the implementation of a new program. Ensuring that the correct version of every piece of software and every document on a CD-ROM is maintained can be a huge headache if this activity is not planned for and staffed. At Hewlett-Packard, we have developed our own configuration management system and electronic warehouse for managing and archiving CD-ROM content. Configuration management systems are also starting to become available for purchase from third party vendors. Be aware, however, that these systems purchases are often the "hidden monster" in launching an in-house production program. The additional hardware, systems, and staff required can add considerable overhead to your operation.

Protection. One of the great advantages of CD-ROM for information distribution is the volume of data that can be distributed on a single piece of media. A problem that stems from this advantage, however, is the need for a security mechanism that prohibits users from accessing any content on the CD-ROM to which they are not entitled. At Hewlett-Packard, we have developed and are using both install-time codeword security and run-time password security, using a

technology called NetLS. Customers are provided with a codeword that allows them to install only those software programs on the CD-ROM to which they are entitled. Once installed, run-time licensing technology built into the software monitors software usage and controls unauthorized copying.

Operating system support for CD-ROM. Keep in mind that the file system format you use to build your CD-ROM applications (e.g. ISO-9660) must be supported on the target operating system platforms. Some operating systems fully support only proprietary CD-ROM file system formats (although this is becoming less and less the case). This issue is further complicated if you intend to produce multi-platform CD-ROMs. In that case, you must accommodate the idiosyncrasies of each platform for which your CD-ROM is intended. (An interesting discussion of some of the issues in producing multi-platform CD-ROMs is available in [2].)

6. Documentation on CD-ROM

What could be so hard about putting documentation on CD-ROM? After all, a mounted CD-ROM looks like any other file system to your computer. Indeed, if your intention is to use CD-ROM to distribute pre-formatted files for demand printing, then you can skip most of this section. However, if your objective is to provide your customers with a high-quality alternative to printed documentation, then there are quite a few challenges in store for you.

6.1 The Case for On-line Documentation

Because the switch to on-line documentation is an expensive and time-consuming endeavor, you will need to present a strong case in favor of undertaking this effort to your management and to the other functional areas that will partner in this effort. In addition to the pros and cons of CD-ROM as a distribution medium, set forth in Section 3.2, following is some ammunition in favor of on-line documentation that has been used successfully at several companies that now distribute on-line documentation on CD-ROM. Also included below are some negatives that you will need to be prepared to address.

- **Competitive pressure.** Providing documentation on-line can still be considered a competitive advantage for many businesses; but within some industries, such as computer hardware and software, where adoption of this technology is growing rapidly, on-line documentation is fast becoming a competitive necessity.

- **Enhanced product value.** Providing software documentation on-line creates a more integrated software package, enhancing the value of the software to your customers in much the same way as with on-line help.

- **Cost savings.** CD-ROM is particularly attractive as an economical alternative to paper distribution. Companies that distribute their documentation on CD-ROM are realizing an order of magnitude savings in associated manufacturing costs.

- **Readability.** Paper is easier on the eyes for extended reading. On-line information is better for information that is "referenced rather than read" [6].

- **Portability.** Paper manuals can be read anywhere, whereas you must be at your computer to read an on-line document.

- **Familiar interface.** People know how to use books. On-line information, on the other hand, is new and can be intimidating to many people.

6.2 Selecting an On-line Browser

When selecting an on-line browser for your documentation, you must choose between two fundamentally different approaches to the display of on-line information: *page-turning* or *soft-copy*. Page-turners attempt to exactly match the look of the printed document, including fonts, display enhancements, and page breaks. The page-turning paradigm says that all information provided by the author through formatting is important and should therefore be retained. Soft-copy viewers, while maintaining the structure of the original documents, do not attempt to exactly match the formatting. The guiding principle behind soft-copy viewers is that information should be displayed on-line in a manner that is optimal for that medium, and that takes advantage of the unique aspects of the medium. Soft-copy viewers generally use scrolling rather than page-flipping, and make much greater use of hyperlink-based navigation mechanisms, such as collapsible tables of contents.

The biggest advantage of page-turners over soft-copy viewers is that page-turning programs usually read documents in their originally authored format. Sun's AnswerBook, for example, reads and displays Postscript. Frame Technology's FrameReader and FrameViewer display Frame format files. Because format conversion is perhaps the single most expensive step in putting documents on-line, using a page-turner as your browser can be a significant time and money saver for your company -- assuming that you do not intend to index your files for keyword search.

While page-turners may appear to offer savings for your company and a whizzy look for your end-users, there are, in fact, some significant disadvantages associated with their use. First of all, because the pages were created with very high-resolution media in mind (paper), on-screen reproductions of these printed pages are often very difficult to read. Further, what looks okay on a high-resolution workstation screen may be completely illegible on a low-resolution PC monitor, and not displayable at all on an ASCII terminal. Page-turners may exhibit slower display performance as well, since pages are rendered on the screen as graphic images rather than as primarily streamed text. The "page-as-image" approach can also result in fewer or less functional on-line features. A page-turner may not highlight keywords following a keyword search, for example. In-line hyperlink features may also be among the missing. In general, it is important not to be overly impressed by marketing hype. Perform usability testing of candidate browsers with some of your potential end-users before making a decision.

Customer-driven feature set. Based on customer research conducted for the most recent version of HP's LaserROM (soft-copy) browser, the following key features were identified as particularly important to end-users:

- **Keyword search** -- full-text search and retrieval capability.
- **Navigation** -- flexible and intuitive ways of navigating through the CD-ROM information base: books organized into bookshelves, tables of contents, end-of-book index, etc.

- **Printing** -- multiple granularities of print functionality: print current section, print multiple sections, print selected text, etc.
- **Graphics** -- legible graphics, either in-line or displayed in a separate pop-up window.
- **Hyperlinks** -- author-provided "jumps" between related topics.
- **Bookmarks and annotations** -- ability to mark locations and attach notes to on-line documents.
- **Customization** -- ability to create customized "views" consisting of subsets of the information on the CD-ROM.
- **Ability to update** -- persistence of customizations, bookmarks, and annotations such that these links are not broken when the CD-ROM is updated.
- **Multiple platforms** -- support for a variety of platforms, including UNIX, PC, and ASCII terminals.
- **Multiple languages** -- localized versions of the browser, including the major European languages and Japanese.

6.3 Text and Graphics Conversion

If you choose a browser that displays all of your documentation in its native format, then you can ignore this section. For the vast majority of companies, however, some amount of format conversion is required: for your legacy data and on an on-going basis. The format you will be converting into is dictated by the browser you have selected. However, be sure that you do not "put the cart before the horse", making the browser decision without considering the consequences of that decision for your data. Remember that your data are your company's "crown jewels". The browser is simply a means for accessing and presenting that data.

SGML. Here is where the value of SGML becomes apparent. SGML (ISO 8879-1986) is a platform- and format-independent language for conveying structural information. An increasing number of companies, including Hewlett-Packard, Silicon Graphics, and Novell, are adopting SGML-based browsers and converting their documents to SGML. (For a good non-technical overview of SGML, refer to [9].) The advantages provided by SGML for on-line documentation and document management include:

- **Separation of formatting and structure.** SGML encodes structure only. This separation of presentation from structure allows the same SGML document to be formatted differently for different viewing environments or applications.

- **Easily localizable.** An SGML-coded document is nothing but ASCII text, and so is completely platform-independent. Further, unlike with page description languages like Adobe's Postscript, or rich-text formats like Microsoft's RTF, the tagging is clearly offset from the text, simplifying the job of localization. Since no formatting information is included in the SGML file, text localization will not introduce formatting problems.

- **Open standard.** Using a standard format protects your investment in your documentation. While authoring systems come and go, and with them proprietary formats, SGML documents are application-independent.

- **Structural views.** Explicit encoding of structure makes SGML particularly well-suited to on-line applications. Using the structure encoded in your documents and an SGML-savvy browser, you can provide structure-based searches (where the search domain is constrained to only selected structural elements within the text), alternate views (such as tables of contents or tables of figures), hypertext, and other added-value features for your customers.

Text Conversion. You have selected (or built) a browser, and it is the greatest thing since sliced bread -- except for the tiny fact that it will not display your documents in their native formats. The biggest headache you will likely experience as a publisher of on-line documentation will be the job of text format conversion. (For a good overview of text conversion issues, and, in particular, issues in converting to SGML, I recommend [1].)

The alternatives available to you for text conversion are outlined below. The code following each entry identifies whether the option can be applied to legacy data conversion (L) or on-going conversion (O):

1. **Author directly** (O) -- If you are in a position to influence a change in the authoring software used by your writers, you may be able to select a tool that generates the necessary format directly. There are several native SGML authoring tools, for example. This alternative eliminates the need for conversion altogether.

2. **Export filters** (O) -- Many authoring tools offer import filters and some offer export filters for some of the more popular text formats. If you are converting between presentation-based formats (versus structure-based formats such as SGML), you may be able to take advantage of these filters. But watch out for unannounced information loss. Most export filters cannot convert all constructs and formatting instructions perfectly, so you will likely need to do some "clean-up" work on the filtered output.

3. **Service bureau** (L,O) -- Service bureaus take your data and a specification of the output format and convert the data for you. Service bureaus are very convenient, but may be costly if you have a large volume of data or need quick turn-around. You also need to keep in mind that, especially if the vendor does not have specific expertise with the target output format, you will need to be very precise in your specifications. Some vendors will offer a reduced price option wherein the gross conversion work is done by the vendor, with the clean-up and exception work left to you. In practice, even if you choose to have the vendor do the whole job, there is usually some amount of fix-up that you will need to do on the converted data after it is returned.

4. **Pre-configured software** (L,O) -- Some conversion software vendors offer versions of their products that are pre-configured to understand input from some of the more popular text processing tools. If you decide to do conversion in-house, purchasing pre-configured software from a conversion software vendor can free you from needing to understand the inner workings of the input format. These same vendors may offer custom development services to further configure their software for your particular output format. The same warning applies here as with service bureaus, however: beware of the results of incomplete specifications.

5. **Software toolkit** (L,O) -- If you prefer to develop your own conversion software, or want to have more control over the maintenance and upgrading of the software in-house, you may find that purchasing a toolkit from a conversion vendor is the best option. Conversion software toolkits usually provide you with a macro-like language and/or API that shields you from many of the nastier and more repetitive aspects of converter programming. Although

this option provides you with a high degree of control, it also requires that you understand both the input and output text formats in detail.

6. **Develop in-house** (L,O) -- Developing conversion software in-house provides you with the highest possible degree of control over the finished product. It also demands the most extensive knowledge of text processing techniques, as well as of the input and output text formats. Unless your application is very simple (very rudimentary input and output formats, for instance), it is usually more efficient and cost-effective to select one of the other alternatives for conversion. Where in-house development often *is* a good option, however, is for automating repetitive clean-up work, in conjunction with one of the other methods for bulk conversion.

Graphics Conversion. Graphics conversion is the easier of the two conversion problems. There are many graphics packages and toolkits, both commercial and public domain, that convert reliably between most of the popular graphics formats. Most conversion service bureaus will convert graphics for you as well as text, although they may themselves out-source the graphics conversion. Things you should watch out for, however, include:

- **Legibility.** Because printer resolutions are so much finer than what can be supported on a computer screen, a graphic generated for paper may not display well on-screen. You may need to manipulate (edit, scale, etc.) the graphic before or after conversion to ensure its legibility on a low-resolution display.

- **Graphics "standards".** Beware: if you have seen one TIFF, you have *not* seen them all. TIFF (Tagged Image File Format) is an open-ended bitmap graphics standard, supported by many on-line viewers, that has about as many flavors as there are fish in the ocean. You need to make sure that the conversion software you select can convert to both the graphic format and flavor supported by your on-line browser.

6.4 Authoring for On-line

Do not under-estimate the impact on-line documentation will have on your authoring community. Unless you involve and get buy-in from your technical publications group early in the process, you will likely encounter nothing but resistance from writers whose perception is that *they* are being asked to do extra work to make *your* project a success. The biggest key to successful authoring for on-line presentation is education. Following are some of the issues you should address in your writer training program.

Authoring Software. If your authors are currently producing documents in a format different from the one required by your on-line browser, then they will either need to start using different authoring software that produces the required format directly, or you will need to set up a new production process that includes format conversion.

Consistency. To facilitate automated (to the extent possible) document conversion, writers must strictly adhere to the styles or templates on which conversion programs are based. The extent to which they deviate from this rule will directly impact the amount of manual work required to correctly convert their documents. Writers must also try to use consistent terminology throughout their documents in order to improve the effectiveness of on-line search for end-users. A glossary of terms can be a helpful tool for authors.

Hyperlinks. While certain hyperlink features, such as tables of contents and indices, can sometimes be constructed automatically by your on-line publishing software, most hyperlinks have to be explicitly added to the document text by your authors. Even the automatically generated hyperlink features typically leverage off of elements added for print (index entries, for example). The requirements with respect to placement and use of such elements are often much more stringent for on-line applications.

Graphics. Figures produced for print may need to be scaled and/or converted for on-line presentation. Authors should keep this in mind and try, as much as possible, to create figures that will be legible at low resolutions.

6.5 The Production Process: In-house or Out-source

Throughout most of this chapter, we have assumed that you intend to publish your on-line documentation in-house. However, for some companies it may make more sense to out-source the production process to a third-party vendor. How do you know if out-sourcing is right for your company? Following are some reasons why companies decide to out-source:

- **Expertise.** For many companies, the biggest hurdle to getting started with an on-line documentation program is the need to develop in-house expertise. Hiring consultants to get you started can help; but ultimately, if your company decides to publish in-house, you will need to have in-house expertise. Out-sourcing can be attractive because it provides your company with on-going access to proven expertise (assuming you work with an established vendor, of course).

- **Economics.** Publishing on-line documentation requires a significant investment in time and resources. The cost of out-sourcing these activities varies widely depending on volume, distribution frequency, customization requirements, vendor selection, and many other factors. Companies often find that their configuration of requirements is such that at least some amount of out-sourcing is more attractive economically than doing the entire job in-house. For example, if you are able to use an off-the-shelf browser without requiring a high level of customization, significant savings can be had on an on-going basis through out-sourcing publishing. Ask potential vendors to include in their proposal a comparison of their quoted price with what it would cost for you to do the work in-house.

- **Commitment to a peripheral activity.** Successful on-line publishing requires commitment across functional areas to the quality of what has traditionally been seen as a peripheral activity -- documentation. Smaller organizations, in particular, may prefer to focus resources on their primary business -- product development -- and farm out documentation production. With the trend toward head-count reduction going on in many industries, this desire to focus on traditional strengths is increasingly becoming true for larger companies as well. Although you cannot get around the fact that your company must take responsibility for documentation content, you can reduce your overall resource commitment (personnel and hardware) by out-sourcing production to a third party.

7. Conclusion

CD-ROM is only beginning to come into its own as a distribution medium. As such, there is both good news and bad news for companies interested in taking advantage of this technology.

The bad news first: getting started can be a complex process. Now the good news: there are lots of people out there who can help you. As we have found at Hewlett-Packard, the most important, and unfortunately often the most under-valued or even over-looked, factors influencing success with CD-ROM information publishing involve establishing partnerships, both internal and external, and gaining cross-functional commitment within your organization.

8. References

[1] Gross, Mark. (1993). Getting Your Data Into SGML. *Technical Communication*, 40(2), 219-225.

[2] Jet Propulsion Laboratory. (1993). Issues in Producing Multi-Platform CD-ROMs. *CD-ROM Professional*, 6(3), 50-53.

[3] Martin, Mike. (1993). Compact Disc Media Evaluation -- What We Now Know About Disc Quality. *CD-ROM Professional*, 6(2), 74-77.

[4] Parker, Dana J. (1993). A Rainbow of Standards. *CD-ROM Professional*, 6(3), 151-154.

[5] The Future of Microsoft. *The Economist*, 327(7812), May 22, 1993, 25-27.

[6] Tynan, Daniel. (1993). Paperless Tigers. *Publish*, March 1993, 49-52.

[7] Udell, Jon. (1993). Start the Presses. *Byte*, 18(3), 116-134.

[8] Weiman, Liza. (1992). How to Make You Own CDs. *MacWorld*, April, 1992, 158-165.

[9] Wright, Haviland. (1992). SGML Frees Information. Byte, 17(7), 279-286.

PAPER NUMBER: 7031

TITLE: Rightsizing Your Mainframe - Performance Criteria

PRESENTER: To be announced
EDI Solutions, Inc.
7760 France Avenue South
Suite #1140
Minneapolis, MN 55435
612-831-9000

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

PAPER # 7032

Rightsizing Your Mainframe Performance Criteria

by James A Hepler
Hewlett-Packard
39550 Orchard Hill Place
Novi, Michigan, 48376-8024
313-380-2207

INTRODUCTION

Rightsizing, Downsizing, or MainFrame Alternative Solutions provide a major opportunity for cost savings while often improving service to the user community. This paper will discuss the predominant issues involved with decision making when approaching a MainFrame Alternative (MFA) situation concentrating on those issues relating to performance. While not all of the issues listed may be directly related to performance, there may be an indirect relationship or an impact on Service Levels that should be considered as part of the overall solution.

MFA covers a broad spectrum of issues. A paper of this type cannot begin to address all of the possible situations or obstacles that may arise. The intent here is to focus at a high level on areas that the analyst will generally have to review in order to size a system, choose a data base, pick a platform, select an application, et cetera as part of the planned main frame alternative solution. Every one of these areas would make excellent topics for other papers allowing more technical detail and more case studies, but a global view is also needed. This paper presents that view from the top.

MAINFRAME ALTERNATIVE DRIVING FORCES

"74% of mainframe users are either investigating, currently migrating from, or have completely migrated from mainframes."

Dataquest 1991

What are those mysterious forces driving computer industry users away from the traditional mainframe environment? There are many factors named in a variety of ways, but they can all be boiled down to a small number. If those are examined closely, you can eventually find a cost benefit. The differentiator is how soon and how easy to measure is the cost benefit?

For the sake of simplicity, let us think of cost savings as short term savings or at least a short term return on investment. Because of technological advances in computer hardware and software, there is an obvious and up front savings in operating costs when a movement from the mainframe environment to a state of the art system and operating environment is completed. But does the cost of migration or conversion to a new environment and the risk to the corporation justify the investment? In about 50% of the cases I have been involved with, this initial cost saving exceeded the conversion or migration investment.

A second driving force is the strategic decision to move to a more Open System environment to take advantage of all the opportunities offered. This is very easy to justify with cost savings as well, but the investment may be higher especially when a move to new Client/Server application technology is to be accomplished at the same time. The return on Information Systems (IS) investment may be farther into the future but advantages to the end users, application maintenance, and operations could be used to accelerate the financial payoff.

This movement to Open Systems is fueled by requirements for features like RAD (Rapid Application Development) and CASE (Computer Aided Software Engineering) tools. Easier access to data for end users and decision support, reduced application maintenance, better networking capabilities, vendor independence, desk-top integration tool sets, better data integrity features, and many other considerations are important facets of the Open System environment.

These types of Information Technology (IT) justifications are based on gaining competitive advantages such as faster time getting new products to market, better product offerings at less cost, more flexible customer service, and so on.

If a company can access their data faster, more intuitively, and with greater flexibility, they can make decisions faster and provide better service to their internal and external customers. This will allow the company to respond more quickly to changing business needs.

Open Systems offers more portability and enabling software so that as technological advances and standards emerge in the future they can be easily implemented. The obstacles experienced moving from the mainframe environment now will be greatly reduced when changes are needed in the future. If the Open System is RISC (Reduced Instruction Set Computing) based, the scalability advantages of RISC will be a large bonus to the user community.

Companies are also examining main frame alternatives as part of a reduction in vendor risk and cost. Many computer hardware and software vendors in the mainframe world are reducing their level of support while increasing the support costs and licensing fees to the customer. Many customers are more concerned about support reduction than increasing costs but it is a problem they cannot ignore. Third party software licensing fees also tend to be larger on mainframes than on Open Platforms.

There is also concern about the future direction of the proprietary product offerings of some vendors by many customers considering an alternative. POSIX compliance is being offered by some vendors as a way of opening their proprietary operating environments but this does not seem to be happening in the traditional mainframe world.

A new trend seen recently is that new college graduates do not want to work in IS departments with tools they consider antiquated or with 3rd generation languages. They prefer to work with new Graphical User Interfaces (GUIs) like MOTIF and Windows and 4th generation languages and CASE tools. A IS staff with a need for entry level programmers and programmer-analysts will need to be concerned with this at an escalating rate.

Perhaps even more critical is the morale problems with existing staff members feeling they are falling behind in technical expertise because of the age of their systems architecture and supporting tools. There is a growing faction of systems personnel who feel that character mode dumb black and white terminals and even PCs with 286 or older chip sets are anchors holding them down. They feel these devices would be more useful at the bottom of a lake where a boat anchor should be. These people may console themselves with the fact that

there are still systems out there with punch cards as their primary source of data entry.

These issues are creating the interest in mainframe alternatives. The benefits can be enormous. There are however many concerns to be addressed when considering a move to a new and lesser known environment. An important criterion in studying the mainframe alternative is to offer the same or better functionality to the users and IS staff while maintaining the same performance service. In many cases, customers will accept a slight reduction in services if there are substantial other benefits or cost savings.

For example, on their previous mainframe system, a customer was getting an average response time of 1.7 seconds. After porting the same application to a less expensive open system the average response time was measured at 1.95 seconds. While it was measurably slower, there was no perception of degraded performance by the users. The system resources were not being taxed and a performance analysis showed that response time would remain at less than 2.0 seconds with 30% increase in usage. This was well within acceptable limits to this customer.

FACTORS AFFECTING PERFORMANCE

There are many factors that can have an impact on system or server performance. There are other factors that can have an effect on performance in a distributed or client/server environment. These can be categorized in five basic categories as follows

- CPU speed and utilization
- Disk I/O rates and demand
- Memory access rates and utilization
- Network delays
- Software locks and latches

The traditional factors of CPU, Disk I/O and Memory Utilization are familiar to most people involved with system performance. The basic concept for these factors is that they are resources required to do work on a computer system and that the amount required of each resource and the time it takes to acquire and use that resource effect response time or throughput. There have been many presentations and papers at Interex covering those topics.

The network delay factor includes not only the speed of the network and queuing within the network, but also delays due to the work performed either on the

remote server or on the client system if one is involved. Stated another way, for a transaction on system A to complete, it may have to wait for part of the transaction to complete on system B. For the local server, this appears to be part of network delay because it takes place on the network outside the local system.

Software Locks and Latches refers to other types of software delays. These can include file locks, contention for data base buffers, messaging delays for local cooperative processing, artificial locks such as for local mail boxes, and other unique delays that do not have to do with the physical resources of the system.

IMPLEMENTATION METHODS

There are five basic implementation methods to meet the business need in a mainframe alternative situation. They are TRANSFER, CONVERT, REPLACE, REWRITE, and SURROUND.

TRANSFER means move the same application from the mainframe to the alternative platform. Many financial and other application packages run on several platforms and can be transferred with a minimum amount of effort. CASE tools, 4GLs, Executive Information Systems, and other types of application software can be ported. Examples of these are Lawson Financials, FOCUS, and SAS. The advantages of TRANSFER include easier transition for program maintenance and little or no training for end users.

REPLACEing the current application with an "off-the-shelf" package offers many advantages. A new package may have a better feature set than a ten year old package or a user developed package. A package specifically designed and tuned for the open environment utilizing client/server concepts will likely offer better response and throughput than existing systems. End users will need to use the new package of course. It is likely that maintenance will be easier because of the lack of "spaghetti" code and newer available tools.

The best example of the CONVERT strategy is to convert CICS/COBOL applications to run in the open environment. This could be an emulation, a coexistence strategy or a movement to C language with a new GUI. There are advantages and disadvantages in this strategy from a performance perspective. Some cases have shown performance improvements, others degradation. The largest advantage is that there are utilities and services that aid in this conversion rather than having to do a total rewrite of the applications.

Automated conversions can be performed by third parties with tools and methodologies developed precisely for this purpose. With this strategy it is also possible to change to a new data base or file structure, to a new user interface, and even to a new language if desirable. For example, customers have converted from DB2 with COBOL and CICS to VPLUS with TurboImage under MPEiX or to COBOL with CURSES and Oracle under HP-UX. Similar to the TRANSFER, the end users would not need retraining and the programmers would be familiar with the source code since it is essentially the same. There may have to be training on the new data base or user interface.

A REWRITE may have the advantage of allowing an improvement in the process or conversion to client/server or adoption of a new data base or file system. There may be a performance edge over some of the other possible implementation paths. With the current Rapid Application Development tools this may be a desirable solution for off-load application targets in particular. End users will have to learn the new system but on the other hand could contribute to a better design. Clearly there would be a relatively massive programming and design effort involved. The expense associated with the REWRITE option is generally high as a result.

The SURROUND strategy is one where the mainframe can sit in the middle of a network of open systems as a server. The mainframe could have a corporate data base so cumbersome that conversion is too complex. With the current newer Middleware tool sets it is possible to use open systems as a client to a large mainframe data base while using the Structured Query Language (SQL) tools available in the open environment. There also may be certain applications that are difficult to migrate that require the mainframe environment. In the purest sense the SURROUND strategy means coexistence with the mainframe while using the enabling tools on the open systems around it. The main advantage to this solution is that the investment in the data base design is protected. Other advantages include the use of SQL type tools and the ability to use less expensive systems for the clients to the mainframe server. The mainframe would be off-loaded because the application would be on the client systems. This could prolong the life of the mainframe and provide cost avoidance for mainframe upgrades.

These implementation strategies have performance advantages and disadvantages in different situations. Deciding whether the solution selected is viable may depend on judgment and knowledge of the performance situations discussed in the sections to follow.

APPLICATION SELECTION

A mainframe alternative may involve off-loading one application or moving the entire suite of applications. Often it is desirable to select one application first. The methodology for selecting this first application varies depending on the needs of the customer. Inherent in any selection must be the acceptability of the resultant response or throughput.

Some criteria for application selection are:
- Query only
- Decision Support
- 3 to 6 month development cycle
- Less than 80,000 transactions per day
- Relatively small disk files
- Packaged applications
- Mission critical application
- Feasible project

The most important thing to consider when picking an application is to be sure the application selected would not be "just a test." If the organization is just "kicking the tires" there will not be enough of an interest or commitment to see the project through to completion. Select one that must succeed.

CPU SIZING

There is no technique for accurately sizing a replacement CPU with a different architecture. The traditional methods use industry benchmarks, marketing information, number of users, number of transactions per hour, MIPS rating, I/O rates, and many other types of metrics that can be used to estimate which CPU can best fit the needs of the application users. Analytic Modeling has been used to study feasibility of migrating and providing suitable performance.

If information on CPU per transaction, disk I/O per transaction, average think time, and number of transactions per hour is known, analytic modeling tools can be applied for interactive applications. For batch type modeling it is necessary to know the CPU and disk I/O per job and number of jobs per hour. Inaccuracy is introduced by not knowing the relationships between the mainframe environment and the new environment. Differences in disk technology and file system access technology must be considered also. Operating system differences such

as internal buffering mechanisms can have a large impact on this type of modeling.

It is necessary to create a ratio of main frame CPU (MFCPU) second to alternate CPU (ALCPU) second to migrate within the model. This is done by using ratios of known performance benchmarks with similar applications or other criteria to estimate the relationship. For example if it is known that MFCPU is rated at 50 TPC-A transactions per second and the ALCPU is rated at 60 TPC-A transactions per second, it is logical to use that ratio for on-line type transactions. It is logical but it may not be accurate.

In this type of scenario, analytic modeling is often used as a sanity check after other CPU sizing efforts using more traditional methods such as comparing similar installed applications at known sites are completed. Other methods are to examine how many users are most customer systems supporting doing similar transactions. It is impossible for any of these methods to be correct 100% of the time. Most analysts responsible for sizing systems in main frame alternative scenarios do it based on a combination of these methods and mixing in a great deal of experience to finally come up with "the answer." Generally a conservative approach is used and if there is any doubt about which CPU is most likely to succeed, the faster CPU will be selected.

It is too simplistic to look at a competitive information chart from one vendor or from an independent third party and say that since the ALCPU is the same speed as the MFCPU it can be a replacement with the same performance. For a first estimate this might be acceptable, but more study is needed. Fortunately it is common for MFCPU systems to have performance reports generated. Unfortunately they are not always accurate.

Another consideration is that if only one application is moving from the main frame, it will be necessary to size based on the part of the main frame that application is using. The trap here is that if the application is using 35% of the MFCPU at a peak time it would be easy to assume that an ALCPU could be used that is 35% the speed of the MFCPU. This probably would not be true. This is the advantage of analytic modeling. The differences in CPU speed per transaction can be considered and the differences in disk technology also can be part of the model. Queues created by waiting for the various resources can be predicted and a resultant response time can be estimated.

In a similar fashion, batch throughput can be estimated. If a batch job takes 8 hours on the mainframe and the ALCPU is 10% faster it could be estimated that

the batch job would run 10% faster. This might be approximately correct, but it is more complex than that. The CPU component of the 8 hours might only be 2 hours and the remainder could be attributed to disk I/O. After migration to the ALCPU, the disk I/O might be 6.5 hours and the CPU 1.8 hours. The total job therefore would now take 8.3 hours even though the CPU is faster. Again this is where analytic modeling may be useful.

Even with analytic modeling tools and services (HPCAPLAN), it is often required to benchmark or do some sort of pilot. Benchmarking is expensive and perhaps only gives an estimate of actual production results. A pilot may reveal unexpected technical issues and is of value as a proof of concept. Often a pilot or small benchmark can be accurately measured and used as a basis for sizing other application migrations. Modeling is a more flexible, less expensive way to make a good business decision on system sizing than benchmarking in most cases. Benchmarking is often more accurate if performed properly however. A good performance consultant can recommend the best solution for the mainframe alternative situation being considered.

DISK CONSIDERATIONS

While there are many disk considerations in mainframe alternatives such as RAID, file space, and mirroring, from a performance point of view the most important factors are the amount of I/O required to do transactions in the alternative environment and the length of time it takes to do an I/O.

Similar to our discussion about CPU alternatives, the alternative platform generally will have disk capacities smaller on average than the typical mainframe with access rates similar, but probably slower. There are exceptions to this guideline however. This means that the disk I/O in an alternative environment may be slightly slower. To offset this handicap, the alternative environments may have better access methods to reduce the amount of physical I/O needed to do the same work. The net of this is that the alternative environment is slightly faster in some cases and slightly slower in others.

The major disk performance concern is generally the I/O system inherent in the data base selected. That will be discussed in the data base section.

There are often operating system requirements for disk space such as a certain percentage of free disk space for optimum performance and adequate temporary and sort space availability. In sizing an alternative solution it is critical to allow for sufficient sort space in particular.

Another area that is often overlooked is limitations to growth in the areas of file systems, total disk space in a single system, and limitations in the data base caused by structural situations There may be a limitation to table size in the alternative relational data base that did not exist in the main frame data base.

Analytic modeling techniques can take this type of knowledge on disk I/O situations and predict both on-line and batch performance as discussed in the CPU section previously. As with CPU, the analysts will take all known factors, mix in any information on similar installed customer systems, and a smattering of experience to come up with a proper disk configuration to allow for acceptable performance. A conservative approach is generally used with disk I/O as well.

Channel and controller speed also can affect I/O access rate when a system is busy enough. This is usually only indirectly considered in analytic modeling but should be considered by the analyst if the I/O rates are high enough.

Disk performance usually starts with, "Will it fit on the spindles?" "Will it fit in the file system?" "Will it fit in the data base?" Sometimes after those questions are addressed, as with CPU, a pilot or benchmark may be needed to see what will really happen with disk I/O. The simplest scenario to predict is the TRANSFER because of the large number of potential changes in the other implementation methodologies.

The SURROUND strategy has a very interesting disk perspective. There are two basic scenarios -- the mainframe contains all of the data and the main frame is a central repository server with distributed data bases on other servers linked in to the corporate data base. The idea is to let the mainframe be the data base I/O engine but let the users use tools on the client systems that give them the accesses they need.

From a performance perspective, we can say we are using less expensive distributed MIPS on the alternative systems while maintaining the corporate data base intact and thus avoiding potential upgrades or poor response on the main frame. This strategy is very important in the very large environments and is a way of improving performance and price/performance while the distributed data base environment is evolving. Part of the SURROUND plan is to realize that long term even the main frame data base will be distributed to alternative networked systems.

In this client/server type of arrangement, the Middleware and the network now become components of the response and should be considered when trying to predict migrated response and throughput. Middleware is software that allows client/server applications to be easily developed and supported.

NETWORKING ISSUES

There are many networking issues with mainframe alternatives. From a performance perspective, we must be sure that the network is adequate to handle the new client/server traffic quickly enough. If there is to be network backup and recovery of files and transactions, the network components need to be able to handle that.

If client/server is part of the direction or application mix, the performance implications of the two tiered or three tiered approaches will need to be considered.

Connectivity and functionality issues abound, but are beyond the scope of this paper. (See the April 26, 1993, issue of COMPUTERWORLD for my article on client/server network performance issues titled "Network Jam".)

Clearly file transfer and applications such as EDI can have a serious impact on system performance and business success. Proper system sizing will need to take these often overlooked system loads into account.

Another possible area for consideration is the concept of overhead and recovery time from two-phase commits from distributed data bases. In the simplest sense, client-server distributed data bases require a two-phase commit for the transaction to complete and this implies overhead on both the client and the server as well as possible network delays.

This becomes even more complex when a three tiered approach is used or when a data base is distributed throughout the network of servers requiring multiple two-phase commits and extremely complex staging of transactions and recoveries when a server is down. These complex recoveries can cause major temporary performance degradation.

It is highly recommended that a network/performance monitor be part of the mainframe alternative solution set. This will help identify and size particular parts of the network that may be slowing data transfers either through high error rates or from too small a pathway for the demand.

Network planning and design are critical to the successful implementation of a mainframe alternative solution. The important thing is to be sure that there is enough band-width for all connections to support the data to be transferred in a rapid enough fashion.

OPERATIONS & SYSTEM MANAGEMENT

Performance monitoring, system tuning, and reporting of service levels are important components of a mainframe alternative solution. Systems need to be monitored for sufficient memory, CPU, and Disk accessibility. Items such as system tables and buffering also need to be monitored and reconfigured as necessary. Often these exercises lead to suggested improvements in the data base or application code.

Software performance engineering techniques are highly recommended if a REWRITE implementation is indicated. Exceptional performance can be designed into the code and the data base rather than attempting to retrofit it later at a higher cost and lower success rate. These techniques can be applied to other implementation methodologies, but the best investment is in new designs.

Consulting may be needed to establish metrics for service level agreements or to implement performance tools properly in the new environment. Without performance tools, the customer is "driving his system in the dark." Reporting mechanisms need to be established as well. These tend to be standard in the mainframe world, but are often overlooked in the open environments.

Capacity planning or other types of long and short range business consulting are also recommended.

Backup and recovery strategies are also important and are sometimes considered performance issues. If the backup strategy selected is too time consuming, operations will be more expensive and may interfere with production work because of the infamous race to daylight.

Is a lights out environment desirable? What about historical or hierarchical storage on optical disks? What about disk mirroring or SPU Switchover to reduce the probability and duration of downtime? These types of facilities are all available in the alternate environments in a different way than they are in the main frame shop.

There are data integrity issues that have an impact on performance. For example if there is to be some type of data base logging, there is certainly overhead in both the CPU and disk I/O areas of the system. This overhead can be as high as 10% extra on what every user does. This factor should be considered in the CPU and Disk configurations as discussed earlier.

DATA BASES

If a data base is transferred, there are two situations which impact performance. Is the data base actually backed up and moved? Or is the data base unloaded and reloaded? If the data base is moved without the unload, chances are that performance will be worse than expected because the data base cannot be tuned for the new environment. If the data is unloaded and loaded into a newly created data base, the performance may be better than expected because of the potential improved design of the data base on the new platform.

If there is a move to a new data base as part of the transition, the performance may or may not be similar. There is a higher chance of accurate prediction if both data bases are relational SQL type data bases for example. If one is networked and one relational, it is difficult to predict performance of an application because of the change in access path to the data.

Selection of a new data base also may be a performance issue. There have been various types of benchmarks done comparing different data bases and even different releases of the same data bases on various hardware platforms. It is important to consider all known information as was discussed in the earlier CPU and Disk sections before selecting a data base based on performance. Current data bases are differentiated more by feature sets than by performance.

The recommended overall strategy should be to select an open data base that performs well and has all the features needed. If there will be a client/server type of access be sure to consider the features and performance implications of that. It also may be wise to review the features and performance factors involved with the distributed data base features of the data bases in making a selection.

Relational data bases have many features and capabilities that can and should be taken advantage of when possible. Some of these items are ability to use raw partitioning, concurrence separation (indexes from tables), striping, spreading data across many less expensive drives and controllers, isolating transaction log

files from other files and controllers, and configuring large memory and buffering areas. Each relational data base allows a different subset of these features.

A very important decision to be made is to choose to repair or improve deficiencies in the current data base structure as part of the migration or conversion. An example of this is a customer with a five year old relational data base. As the data base evolved and needs changed, the staff did not take the time to alter the data base table structure to meet those needs the most efficient way. Instead the expedient way was found to add new tables, creating redundant data structures and wasting I/O and CPU as a result. As part of this customer's conversion to a new relational data base, the integrator redesigned the tables and the 4th Generation Language access routines to repair the problem. This allowed the performance that resulted to exceed expectations.

It may be very wise to purchase some data base design and tuning consulting from the data base vendor as part of the implementation plan and to allow time for those functions in the project.

THIRD PARTY APPLICATIONS

Like the CPU and Disk discussions earlier, if a third party application is selected to replace an existing one, examine whatever data is available including vendor information and benchmarks as well as any customer reference sites for information on expected response. Data base used also will be a part of this decision process.

If an application is to be transferred, try to be sure it is a version optimized for the new platform rather than a port of the source code. If you have to choose between one that is ported or one that is designed and written on the new alternative system, pick the native version. There is a high probability the performance will be better.

Be conservative in your expectations. However, it has been my experience that the newer versions of these applications tend to run better than the old main frame versions.

CLIENT/SERVER

If possible, move to client/server applications as part of the conversion effort. Often this is a REWRITE, but it also may be a REPLACE strategy. The performance implications of client/server are that cheaper MIPS will be doing the

work with similar response in most cases as long as the network is adequate. There are also all the other benefits associated with client/server technology such as the better tools and lower maintenance.

With a move to client/server, the expensive MIPS will be saved to be used for other things. In the SURROUND situation, the main frame life without upgrade may be extended. In the REWRITE scenario, the new server will have a similar benefit and may be able to be a smaller CPU than was originally expected or sized based on a main frame CPU to alternate CPU comparison.

It is very common to suggest a move to client/server as phase II of a migration project. This sounds like a good strategy, but in real practice it seldom gets completed. Phase I is of course to get as much off the main frame as possible to save money and gain the other benefits discussed. It is wise to size the system for phase I and not for phase II that may or may not be in the future.

CONSULTING

In addition to the network, data base, and performance consulting mentioned earlier, migration planning and project management consulting may be necessary to implement the project in a timely fashion and to be sure that the performance needs and milestones are not overlooked.

Integration and conversion consulting also may be needed. Be sure that all consulting is scheduled in the project and meshes with the requirements definition.

SUMMARY

There are many performance considerations in planning a main frame alternative solution. This paper has attempted to discuss some of them at a high enough level to be applicable to a variety of situations. In whatever situation develops, some or all of these topics may come into play. The analyst or consultant must gather all the information available and proceed with the best business decision possible. It is critical to the successful completion of a main frame alternative project that consideration be given beforehand so that surprises are minimized and that performance service levels are met.

Paper 7034
Cooperative Design Methodology
Developing a System with End Users as well as For End Users

By Pamela Herbert
Octet Consulting, Inc.
Walnut Creek, CA.
510-942-3027

In the traditional development of systems it is not uncommon for managers from the data processing department to work with managers from the user community to design a system that handles some particular business function. There is an inherent short coming in this methodology in that the actual end users of the system, having had no voice in its creation, often don't like it. Typically a system developed this way is uncomfortable if not awkward to use and is often missing certain details that then have to be handled manually or through the use of cumbersome work-arounds. The end result in the implementation of these systems can be user dissatisfaction leading to low morale and reduced productivity. It is also very expensive to design an incomplete system and then to have to retrofit the system to include features that were unforseen in the original design due to a lack of understanding of the end user's job.

There is a better way to design a system. A data processing system developed in cooperation with its users is much more likely to have the features needed by the users of that system and to perform all of the requisite business functions. Getting end users involved in the design process is still a fairly novel idea but is one well worth examining. Reluctance to adopt this methodology is due at least in part to the misguided notion that end users who have no formal training in systems design have nothing to offer in the design process. There are, in fact, several advantages to using a cooperative design methodology. Three of the major advantages are:

- The synthesis of a diverse expertise in solving the problem produces a fuller more correct solution.

- Participation in the design process engenders full ownership of the system by those who will be using it. This is especially critical when it comes to implementation time where bugs, quirks and oversights rear their ugly heads. Users who feel that the end result of the implementation was at least partly their doing will be far more accepting of these problems then those who feel they have unwillingly had the system foisted upon them.

- Giving the end users an opportunity to participate in decision making always

has a positive effect on morale. People like to feel that their voices are being heard when it comes to how they do their jobs.

- The cooperative design process provides a forum in which the system developers can get constant feedback as to how they are doing to satisfy the end user's needs. (This is both useful and dangerous as will be discussed later.)

For the purpose of illustration I will use the development of an order entry system because it is an area in which I have some expertise and one most anyone who has ever ordered anything can relate to.

Certain assumptions have been made concerning this system. In our company we develop systems in a Fourth GL using an iterative prototyping methodology. This can be very useful since the users aren't always able to articulate their needs until they have tried using the system. It also makes it possible to try out different approaches to solving a problem and having more designers generally leads to the proposal of more approaches. This paper assumes that you begin the cooperative design process after management and the systems professionals have come to some conclusion regarding the general approach to and scope of the project. A budget may already have been agreed to.

I feel the need to apologize at this point for referring to the users of the system as 'end users'. In this day and age this is considered by some to be politically incorrect. Information management professionals are now encouraged to refer to the actual users of the systems they build as 'clients' or 'customers'. I agree with this in theory and even in practice however, for the purpose of this paper I think that 'end user' is an accurate description and therefore acceptable. All persons are referred to in the male gender.

THE VALUE OF COOPERATION AND ARTICULATION OF MUTUAL EXPECTATIONS

Fundamental to this design approach is a recognition of the value of the cooperative effort. "Virtually all work has a cooperative element. To take an extreme case, even prisoners in chain gangs often cooperate in some simple ways with their guards in their conduct of daily life. Otherwise, they would have to be dragged from place to place, even if they did not rebel violently. If, however, their relationships were primarily cooperative, the prisoners would not be chained."[1] A work place in which users are coerced to accept a system they find unacceptable can have a very punitive feel about it. In developing a new system it is wise not to cast the systems professionals into the role of prison warden, delineating all of the rules and regulations and then enforcing them. It is, however, the job of MIS management to ensure that once mutual goals and expectations have been set, the team adheres to mutually arrived at budgets and schedules.

A developer's goal in creating a new system, simply stated, is to provide full business functionality and increased productivity in a cost effective manner. An end user's goal is to build a system full of bells and whistles that makes his job more efficient and easier to do. These are not mutually exclusive goals but without a concerted effort to keep them both as stated end points of the project they can cause friction. It is critical that the project be managed with both of these objectives in mind or both won't be met. Instead, the end product will be either a system that comes in on time and within budget but doesn't fit the user's needs, or a fabulous system that the users love but that is late and over budget. Neither outcome is acceptable. This paper is not an attempt to outline a detailed methodology for producing a system that meets both goals but rather a description of the elements required to effect cooperative design with some discussion of project management issues that must be attended to in order to avoid disaster.

THE STARTING POINT FOR COOPERATIVE DESIGN

The first thing that must be done is to decide who will be on the development team. At the very least the team will consist of the information management project manager(s), the business manager ultimately responsible for the system (in this case the operations manager) and the managers of the end user departments. Including the end users at this point is optional as the department managers can garner end user support and willingness to participate in a departmental meeting later on.

Once the development team has been assembled, have a kickoff meeting to discuss the goals and scope of the project. It is in this meeting that the entire team will define goals and expectations for the project and establish what the end user's goals and expectations are. MIS can then acknowledge its understanding of the user's goals, describe the methodology to be used to meet their needs and the ways in which end users will be included in the design process. By the end of this meeting the business managers and any other users involved should begin to feel like members of the development team and will be in a position to go to their employees and/or peers to recruit them in a positive, affirmative way. Having this meeting is critical in order to garner the departmental

manger's full support for dedicating their employee's time to the design effort.

One other key player on the team has to be an upper level manager who can function to ensure that both MIS and the business unit carry out the project plan as it is defined. This high level champion is essential to ensure success since neither the MIS manager nor the business manager are in a position to 'watch dog' each other when problems arise. Without a single, high level overseer, the project can be stopped dead at any point due to lack of performance (either intentional or not) on the part of any other manager on the team. This person must be fully committed to the collaborative effort and function from an unbiased perspective.

THE VALUE OF EACH PLAYER'S PERSPECTIVE

How can an order entry clerk actually participate in the design of an order entry system? A better question would be how can you design a decent system without the full expertise of the order entry clerks? Without the detail knowledge these people have about what it takes to get from a phone conversation with a customer, to shipping the correct merchandise out of a warehouse to the proper address, the system designers simply can't design a usable system.

System designers understand conceptually what the pieces to an order entry system are. They understand the file management system being used and its features and limitations. They understand the programming language being using and how it interacts with the screen handler and the file management system. They also understand information processing, data flow, perhaps a little system management and details about how the current system works from a data processing perspective (if there is an existing system). None of this means that there is a full understanding of what must be done to produce an order entry system for these particular users to handle this particular business.

The order entry clerk understands exactly what it takes to translate the customer's needs into an order using the current system. He understands how to get the information out of the customer that will allow him to figure out the correct item number, how to come up with a price, how to communicate this information to the warehouse and what procedures to use when there are problems filling the order. None of this qualifies this person to design a system to manage the flow and collection of information required to carry out the stated business functions (including sales history and service level statistics). It is only through the intelligent combining of these two areas of expertise that the correct design for this order entry system can be derived.

MUTUAL LEARNING [2]

Successful cooperative design requires that both the end users and the systems designers feel that they are benefiting from the relationship by learning from each other. This means that the systems developers learn about the business function and the end users learn about the technologies that can be used in automating their jobs.

There are two major techniques to be used by systems developers to gather the detailed information concerning the user's job. In the initial phases of system analysis the systems designer will need to interview the users. The immediate goal of these interviews is to get a clear picture of the user's day to day tasks and the precise, current procedures used to carry out these tasks. With some users it is sufficient to ask them to describe their job in detail. Not all people respond to such a vague request, however. It is not uncommon for a user to view his job as simplistic, common and not requiring explanation. This type of person approaches the task of describing his job with an attitude of 'everybody knows that, don't they?' and assumes no explanation is necessary for the most fundamental, and probably most crucial aspects of his job. If, as an interviewer, you find that you aren't getting more information than you can easily process you need to ask more specific questions. Sometimes a question like, "what goes well in your job? What are some effective procedures you use" or "what is the most difficult/cumbersome aspect of your job?" can lead the user to impart important details more easily.

Secondly, the systems developer must observe the user doing his job. This observation should be passive and quiet without a lot of interruptions to ask questions. Again, the objective here is to get a good high level view of the user's job. Make notes concerning questions or manual procedures and pursue these areas in further interviews. Clearly, at this point, the systems designer is doing all of the learning since the user already knows his job.

The order taker will be much more useful as a design ally if that person is taught enough about the technology being used to allow him to assist the designer in applying that technology to the system design. For example, the business may be moving to a client/server system that has a graphical user interface (GUI). The users may already be New Wave or MacIntosh users in which case they will have some sense of what the technology can do and may have some ideas as to how they would like it applied in the new system. If, however, they have never used a GUI based system they will need to be taught what it is and the ways in which it is useful. An interviewer may say to the user, "We could provide you 'button' to click on near the description field and, after entering an item number, 'pushing this button' would cause a window filled with other types of detailed information concerning the item to 'pop up'. Would this be useful for you? How else might you like to see this type of feature used?" If the user has never used a GUI before, this description will be quite cryptic and some other GUI application or a prototype application will have to be shown to the user. Once the user knows what is being described, he may be able to suggest other areas in the system where the use of that technology would be beneficial.

NEW WAYS OF THINKING

It is critical to understand that the use of new technologies to solve existing problems can have a tremendous impact on the methodology used to solve those problems. In other words, the introduction of a new system using new technologies can actually alter they way work flow is handled and change the very nature of the user's job. A systems designer is more likely to conceptualize these changes and it is the designer's job to draw out of the users those areas that are most in need of change due to a changing business climate.

Joint design sessions should focus on concrete applications issues. The end user does not consider his job in terms of information flow and management. To him there are a number of tasks that need to be accomplished and he may even have a very strong idea as to the correct procedure to use in solving the problem. It is the job of information managers to raise the user's level of thinking to higher ground in order to enable him to accept or even propose alternative solutions to the problem.

Consider the replacement of a multi-part form with an on-line function. Specifically, in one system we developed, the users had been using a Problem Information Form (PIF) to document, communicate and record the resolution to all problems related to order entry. Problems could span multiple functional areas including customer service, transportation, inventory management and accounts receivable. Problems often began their journey in the customer service area when a customer called to complain about an order or when an order taker couldn't fill an order due to inadequate inventory. The customer service representative (CSR) would take a blank PIF, check off the appropriate boxes, write in some description of the problem and what he had been done about it so far, and route it to the appropriate department for action and resolution. Once it got to its destination the user there might discover that the problem really originated in another department so he would make additional notes and send the form on its way. If the problem was particularly urgent and the appropriate person was not at his desk, a highlighter pen was used to highlight the entire page and the CSR taped the form to the person's terminal. This was a very low tech, but unquestionably high visibility system.

Additionally, these PIFs were used to track customer service statistics for performance measurements. Information concerning fill rates, inventory control, carrier reliability, and customer satisfaction was extracted based on the boxes checked on the form and the verbiage manually entered. This tracking was done manually by a person reading through the forms, categorizing the information and entering it in to another file by category.

We replaced this system with an on-line function that accomplishes the same goal. Had we simply looked at the form and designed a screen that simply resembled it, our design would have failed because it would not have include a communication mechanism for routing the PIF on-line. Our ability to categorize the problems for statistical reporting might also have been inaccurate because there is a human interface between the actual data recorded on the form and the data entered in to the PIF statistics data base. We needed a complete understanding of all the ways in which the PIF is used and the user needed some understanding of the technological possibilities of putting the PIF on-line in order that we

jointly develop a full solution to the problem.

The cooperative design process is evolutionary in nature. Our initial design proposal for the PIF, based on many hours of interview with order takers, transportation clerks and AR clerks, had some features that the users found to be very attractive. For example, for PIFs related to orders the system automatically fills in the order information (order number, trucking company, customer, shipping address) with no effort on the part of a user. They felt uncertain, however, about the communication mechanisms we proposed. They were very accustomed to using highlighting pens and tape to convey urgent messages concerning problems. Our sense of alternative communication methods (e-mail, phone) didn't satisfy the users' sense of their needs. Because we included them in the design process and showed them a new PIF methodology in a prototype we were be able to arrive at a better solution before the system had been implemented in a production environment. This enabled us to modify the system in accordance with their needs without incurring all of the headaches of making changes to a production system and without forcing them to use a technology they felt uncomfortable with when they were actually trying to do their jobs.

WHAT MEASURE, SUCCESS?

The end users are never aware of the cost of their wishes and they aren't often persuaded to put budgetary concerns ahead of their needs. For them the success of the system is a function of how many of the features they requested are actually in the delivered product. Upper management has a manifest to hold all departments to stated budgets and to keep business flowing at a smooth and reasonable pace. It is therefore critical to upper management that new systems get delivered on time and within budget. Project managers are charged with the duty of fulfilling both of these goals.

Generally, by the time the users have been included in the design process a proposal has already been submitted to upper management outlining the scope of the project and a proposed budget. After including the users in the design process new system needs will be uncovered that were not included in the original estimate. Resolving this discrepancy requires very careful management. If it's not possible to get approval for an extended budget, new features will have to be designated as being out of the scope of the project and not included in the delivered product. It is crucial that the users understand that if a feature they want is not included in the final product it is because it was designated as out of the project scope and not because the information management people decided that it wasn't really necessary. To the greatest extent possible, the users need to be encouraged to agree with the perspective that making such cuts is in the interest of furthering the business and making as many improvements as possible in the shortest possible time frame.

Although deadlines and project budgets can appear to be frustrating limitations on the system developer's ability to satisfy the user's requests, they can also work to the project's advantage. A system that isn't implemented until well past the stated release date due to constant pre-release enhancements and that goes substantially over budget loses credibility in everyone's eyes. Even though the effort to implement all possible enhancements is an attempt to fulfill the user's dreams, when these efforts cause the

implementation date to be extended, it appears to the users that project leadership has made commitments that can't be kept. This generally leads to the entire project becoming the object of much negative speculation. Upper management's perspective on over extended budgets is well known.

It is to everyone's advantage to implement an initial release of the software on time and within budget and to use the success of that implementation to get further funding for a subsequent release that will include features not yet implemented. This is a win/win approach for everyone.

CONCLUSION

We have found that using a cooperative design methodology results in a solid system that is quickly and easily accepted by the end user community. No methodology is without pitfalls and in this case one of the biggest threats to success is to have the users eagerly expecting to have their design ideas implemented only to find them excluded from the software for budgetary reasons. With proper management and a very strong effort to inform the user base of the precise scope of the project this risk can be minimized and all of the players in the production of a new system can have their objectives successfully met.

References

1. Kling, R. Cooperation, Coordination and Control in Computer-Supported Work. Communications of the ACM 34,12 (Dec 1991), 83-88.

2. Kyng, M. Designing for Cooperation: Cooperation in Design. Communications of the ACM 34,12 (Dec, 1991), 65-73

Interex

P.O. Box 3439
Sunnyvale, Ca 94088-3439
(408) 747-0227
Fax (408) 747-0947