PROCEEDINGS OF

THE INTERNATIONAL SOCIETY OF PARAMETRIC ANALYSTS

FIFTH ANNUAL CONFERENCE

ispa INTERNATIONAL SOCIETY OF PARAMETRIC ANALYSTS

VOLUME II NUMBER I

ST. LOUIS, MISSOURI
APRIL 26–28, 1983

A METHOD TO MEASURE THE "EFFECTIVE PRODUCTIVITY" IN BUILDING SOFTWARE SYSTEMS

by

Lawrence H. Putnam
Douglas T. Putnam
Lauren P. Thayer

Quantitative Software Management, Inc.
1057 Waverley Way
McLean, Virginia 22101
(703) 790-0055

A METHOD TO MEASURE THE "EFFECTIVE PRODUCTIVITY" IN BUILDING SOFTWARE SYSTEMS

We have been building software systems in the U.S. for over 25 years. The computer industry has not found many, if any, good methods to quantify and measure changes in the overall efficiency of building software systems.

Generically this is referred to as "productivity". Yet productivity, at least the traditional definition (Total Output/Total Effort), appears to be inadequate and a counter-intuitive measure for software systems. The productivity metric is not adequate as a standalone measure, but it can be used with a combination of metrics related to resource consumption. By using a set of integrated metrics the software measurement process will be more complete with respect to the management information that needs to be quantified.

PRODUCTIVITY - AN INADEQUATE MEASURE BY ITSELF

First we need to explore what productivity is, how it should be perceived and why it is inadequate as a standalone measure.

In keeping with the traditional definition, productivity for a software system is the total output divided by the total effort required to produce the output. The total output of a software system is the functionality that is created. This might be thought of as the final end product. It is proportional to the number of "developed" executable lines of source code or any lines of code that required work.

It is very important to recognize that this ratio measures the productivity for the system. Many people mistakenly think of system productivity in terms of programmer productivity. This is a dangerous conceptual problem because programming effort is only a small portion of the work involved in building systems. A great deal of work is expended in the front end creating a design. Later in the process a very significant amount of effort is used integrating and testing at the unit, subsystem and system levels.

The most serious problem with the productivity ratio as an objective measure is its extreme time sensitivity. Notice that time is not explicitly stated anywhere in the definition. Rather it is implicitly buried in the denominator term (manmonths of work). A better definition that separates time and effort is needed. Time and effort are related but in a non-linear way. A powerful exponent attached to the schedule causes the manmonths to change drastically as the schedule changes modestly. If the manmonths are so sensitive to changes in time then productivity is equally sensitive.

There are a number of different ways to approach building a given software system. One extreme uses a very large number of people to get the system built in the shortest possible time. The other extreme is to let a few people work for a longer time until the job gets done. This situation usually happens when people or money are scarce. Keep in mind there are a large number of different time-effort strategies between these two extremes.

If productivity varies with changes in the planned schedule then the productivity can change dramatically from one development to the next. Therein lies the problem.... is it possible to measure systems effectively with a metric that is so sensitive to the schedule?

A SET OF MANAGEMENT METRICS

This paper presents a set of measures that can cope with the sensitivities of the software development process. A sound theoretical basis will tie these metrics together in a consistent and reinforcing way. The metrics that will be used are:

1. Technology factor (A Macroscopic Efficiency Index)
2. Productivity (Source Statements/Manmonths)
3. Average Manpower (Total Manmonths/Time)
4. Project Duration (Schedule)
5. Total Manmonths (Work)
6. Mean Time To Failure (Reliability)

These measures will tie productivity, resources and quality aspects together to provide a quantitative basis for effective strategic planning, management control, and good software decision making.

The Software Equation (Trade Off Law)

Seven years ago when Larry Putnam was working for the Army at the Computer Systems Command at Fort Belvoir he discovered an algorithm known as the Software Equation.

$$S_s = C_k \cdot K^{1/3} \cdot t_d^{4/3}$$

where,

S_s = The number of developed executable source statements

C_k = Technology constant

K = Life cycle effort = $\frac{E}{\beta} = \frac{\text{Development Effort}}{\text{Adjusting Factor}}$

t_d = The development time (schedule)

β = Adjusting factor

The software equation states that for a system of a given size, it will take t_d months using E manmonths of effort at an efficiency level of C_k. Note that time and effort are multiplied together. A change in the schedule will cause a change in the effort required to get the product done to meet that schedule. The software equation can be thought of as a software trade-off law since it is possible to trade time for effort.

This is an extraordinarily time sensitive process. Very small changes in time produce very large changes in effort. The software equation is very powerful. If an organization deliberately plans to take a little longer they can reduce the effort required to produce a system by a very substantial amount. However, this policy must be adopted before the start; you cannot slip into it. A few good people given enough time to build a product right will create fewer errors that will have to be found and fixed later. Most people familiar with software development agree that there is a time-effort trade-off relationship. The magnitude of the trade-off is sometimes questioned. A substantial amount of data supports that it is close to an inverse 4th power time trade-off.

The Technology Constant (An Efficiency Index)

Is there any way the software equation might be able to quantify the development process in an all inclusive way? The technology constant (C_k) in the software equation is quite well suited to this task. The technology constant is a macroscopic parameter that measures the aggregate impact of all influences that effect how long it takes and how much effort is required to build a system. In essence this parameter measures how efficient an organization is in building a certain class of software. For example, a very efficient producer of MIS software would have a high value for C_k. This type of environment would typically include on-line development, good tools, strong professional skills, stable requirements, etc. A less efficient producer would have a lower value for C_k. Higher values of C_k will produce shorter schedules that require less effort for a product of a given size. The real objective in building systems is to produce an equivalent or better quality product in less time using less effort. The technology constant measures these objectives very well.

After collecting and analysing a large database of technology constants (over 800 systems) values for C_k have been established that represent a baseline for different classes of software. The baseline values are primarily driven by the complexity of the particular class of work. The same values of C_k have been observed in similar organizations building the same type of software during the same time frame. It appears that it is the best unambiguous measure of the overall effectiveness of building software found so far. It is important that one deals with a homogenous class of work in making their measurements (i.e., all business systems, or all real-time embedded systems, etc.)

How does one get the value of C_k? For a system that has recently been completed the following information is known:

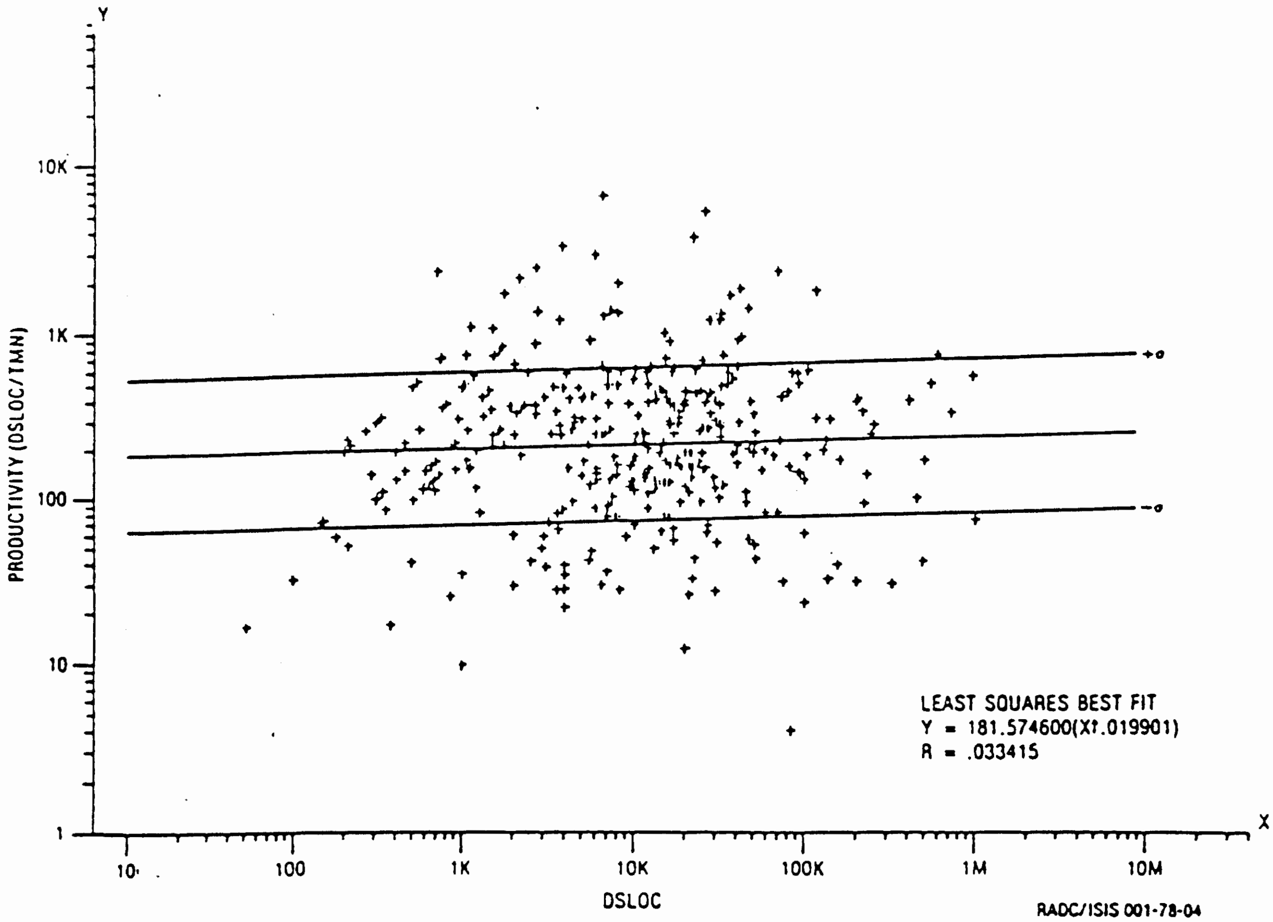
1. The number of developed lines of code
2. The development time
3. The development effort

With values for three terms of the software equation it is possible to calculate the fourth term, C_k.

Establishing Trend Lines for Metrics on Resource Consumption

There is still a need to bring measures of resource consumption into this process. The U.S. Air Force spent several years collecting a large quantity of data on software projects . This is known as the Rome Air Development Center (RADC) database. It is a very good collection of software data from a wide variety of applications and organizations. This is the largest heterogeneous database ever collected. Richard Nelson of RADC analyzed these data with respect to the management parameters - size. schedule. effort. average staffing and productivity.

These are shown as Figures 1 - 4.



Productivity vs. DSLOC.

Figure 1.

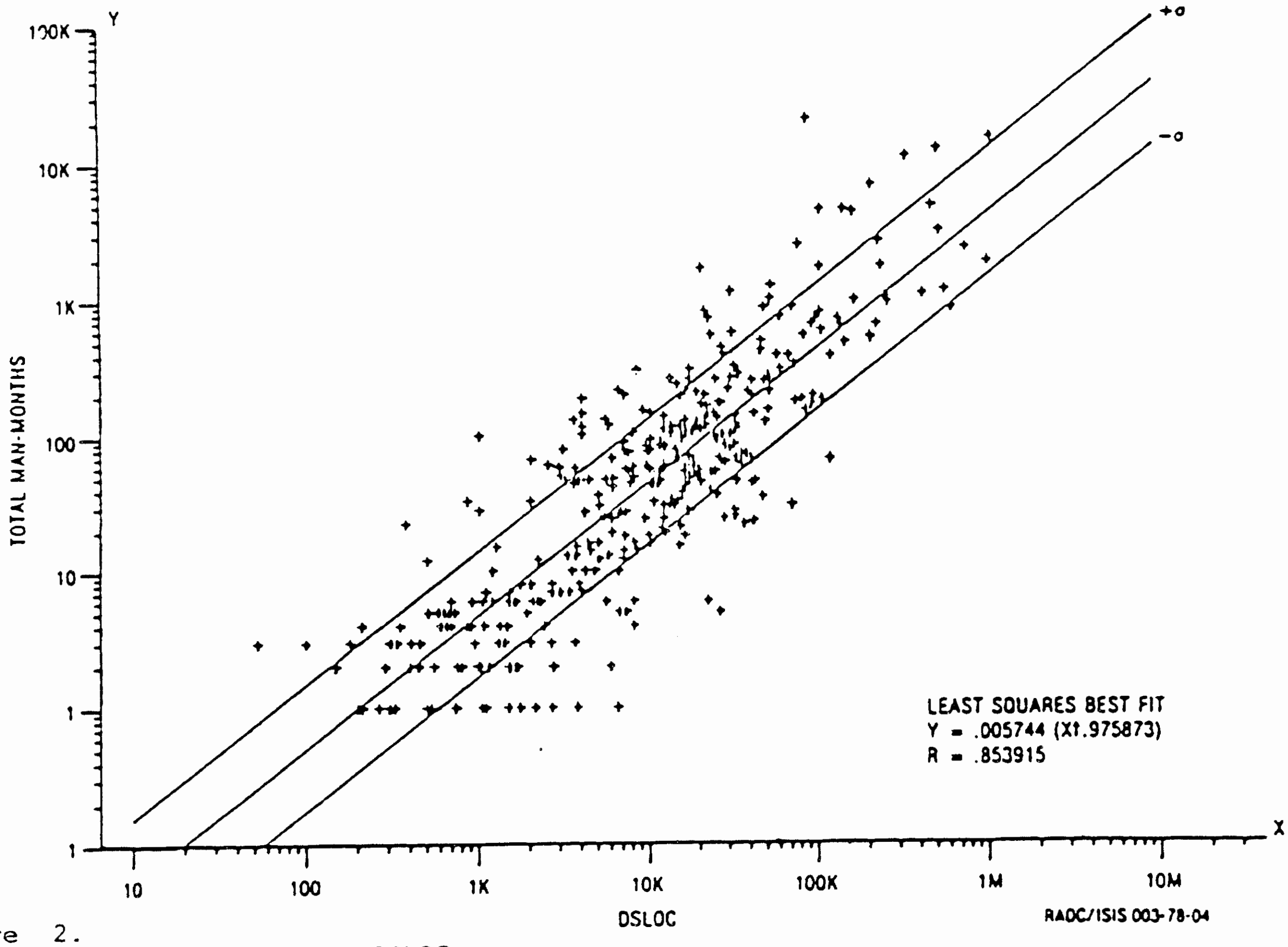
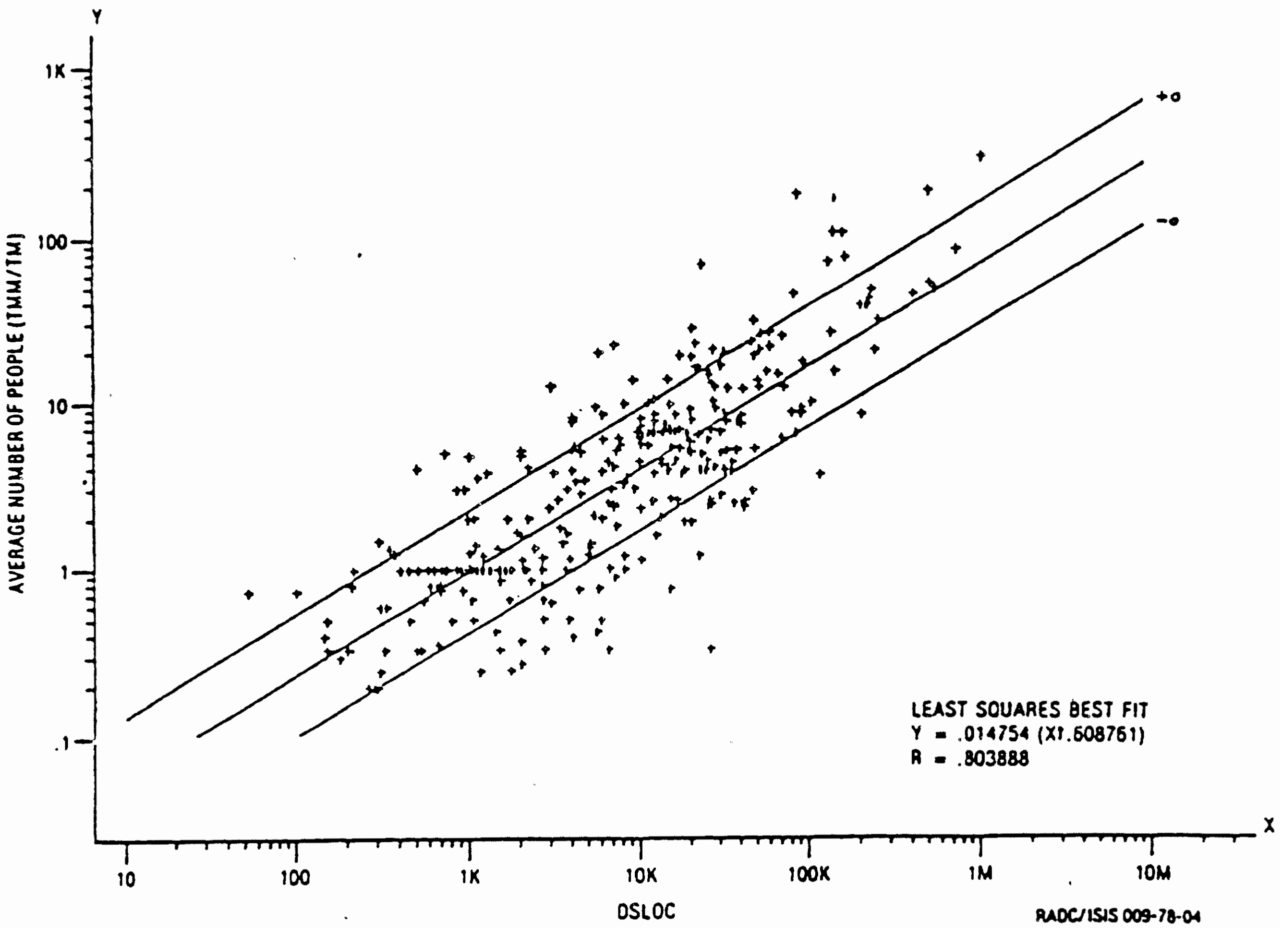


Figure 2.

Total man-months vs. DSLOC.



Average number of people vs. DSLOC.

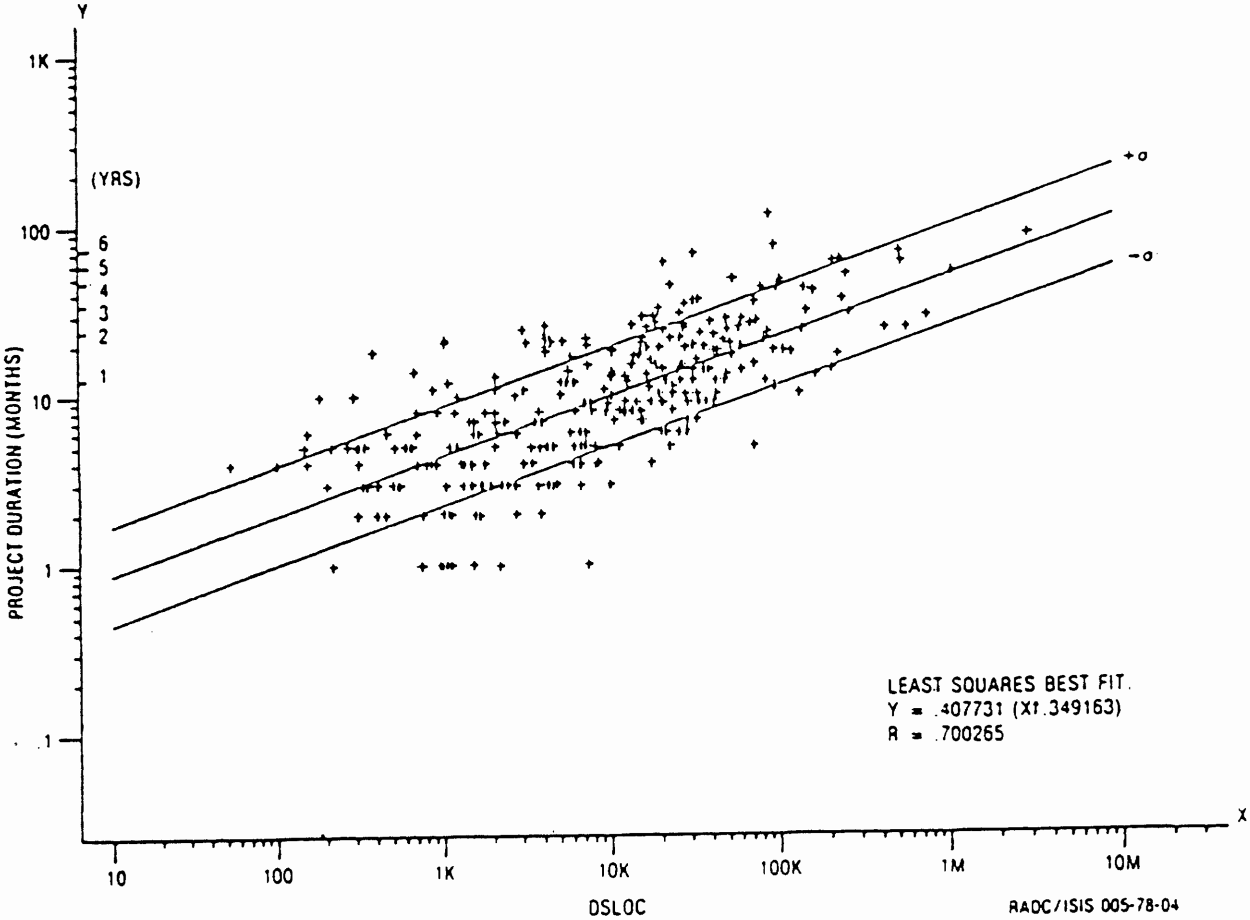


Figure 4. Project duration (months) vs. DSLOC.

On each of the RADC graphs the horizontal axis represents source lines of code. Note that both scales are logarithmic. Power functions plot as straight lines on logarithmic scales. The data ranges from 10 to 1 million lines of code which spans the range of most systems. The plotted lines show a best fit of the data for productivity, project duration, total manmonths (proportional to cost), and average manpower. The middle line in each case represents the average best fit of all the data points. The upper and lower lines are the +1 and -1 standard deviation lines.

The correlation of the trend lines are quite good on all the graphs except productivity. We know that productivity is very sensitive to the schedule and this is why the correlation is close to 0 (no fit). Even though the trend lines on the other graphs look good there is still a large variation in the data at any particular system size. For example, at a system size of 10,000 lines the project duration could be from 1 month up to 1.5 years. If there is some way to discriminate within these data sets then they can be used in a comparative way. The technology constant gives us this capability. Notice that the information used to plot the data points on the graphs can be derived from the inputs to the software equation. With the additional sorting capability provided by the technology constant the four graphs will be more meaningful.

The RADC trend lines appear to be valid. A new set of data superimposed on the trend lines will give a new basis of comparison. From this analysis it will be possible to evaluate the overall effectiveness in management terms. This tells a complete story about the development philosophy and management style of a software organization.

CASE STUDY OF THREE SYSTEMS

To demonstrate this method it is worth looking at the data for five real systems built by three well known and respected organizations. This case study will look at three distinct classes of software. The three application groups are:

1. Real time embedded software
2. Systems software
3. MIS software to support manufacturing operations

The real time system is the software for a cruise missile. The systems software project is called EXEC System and can be classified as an operating system. RFM, Parts No., and Materials are MIS systems and were developed by the same manufacturing company.

Shown in Table 1 are calibration runs using SLIM* to calculate C_k with a computer.

---

* SLIM is a computerized implementation of the Putnam software equation and the Norden/Rayleigh resource allocation model. The Calibrate function calculates values of C_k from data of previously developed software projects.

Table 1.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                              CALIBRATE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

SYSTEM NAME: CRUISE MISSILE

SIZE (SOURCE STATEMENTS): 5800

DEVELOPMENT TIME: 24 MONTHS

DEVELOPMENT EFFORT: 107 MANMONTHS

GRADIENT LEVEL: 1

TECHNOLOGY FACTOR: 1

$C_k = 754$

THIS TECHNOLOGY FACTOR SEEMS UNREASONABLE. PLEASE RECHECK YOUR DATA.

SYSTEM NAME: EXEC SYSTEM

SIZE (SOURCE STATEMENTS): 61000

DEVELOPMENT TIME: 33 MONTHS

DEVELOPMENT EFFORT: 248 MANMONTHS

GRADIENT LEVEL: 1

TECHNOLOGY FACTOR: 8

$C_k = 4181$

SYSTEM NAME: PARTS NO.

SIZE (SOURCE STATEMENTS): 108000

DEVELOPMENT TIME: 21 MONTHS

DEVELOPMENT EFFORT: 25 MANMONTHS

GRADIENT LEVEL: 1

TECHNOLOGY FACTOR: 16

$C_k = 28,657$

SYSTEM NAME: RFM

SIZE (SOURCE STATEMENTS): 100000

DEVELOPMENT TIME: 21 MONTHS

DEVELOPMENT EFFORT: 48 MANMONTHS

GRADIENT LEVEL: 1

TECHNOLOGY FACTOR: 15

$C_k = 21,892$

SYSTEM NAME: MATERIALS

SIZE (SOURCE STATEMENTS): 700000

DEVELOPMENT TIME: 38 MONTHS

DEVELOPMENT EFFORT: 384 MANMONTHS

GRADIENT LEVEL: 1

TECHNOLOGY FACTOR: 17

$C_k = 35,422$

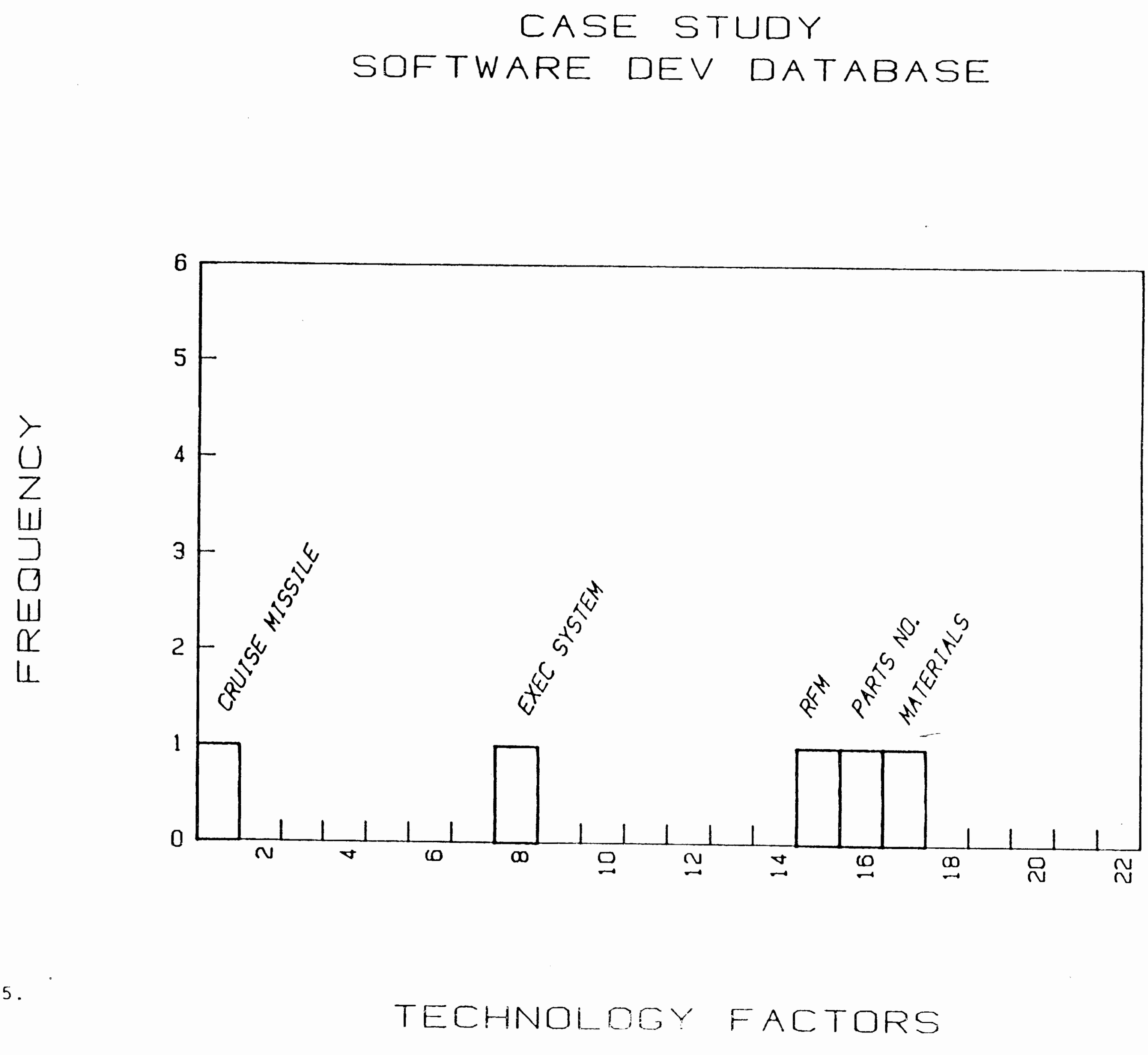

CASE STUDY
SOFTWARE DEV DATABASE
FREQUENCY
0
1
2
3
4
5
6
CRUISE MISSILE
EXEC SYSTEM
RFM
PARTS NO.
MATERIALS
2
4
6
8
10
12
14
16
18
20
22
TECHNOLOGY FACTORS

Figure 5.

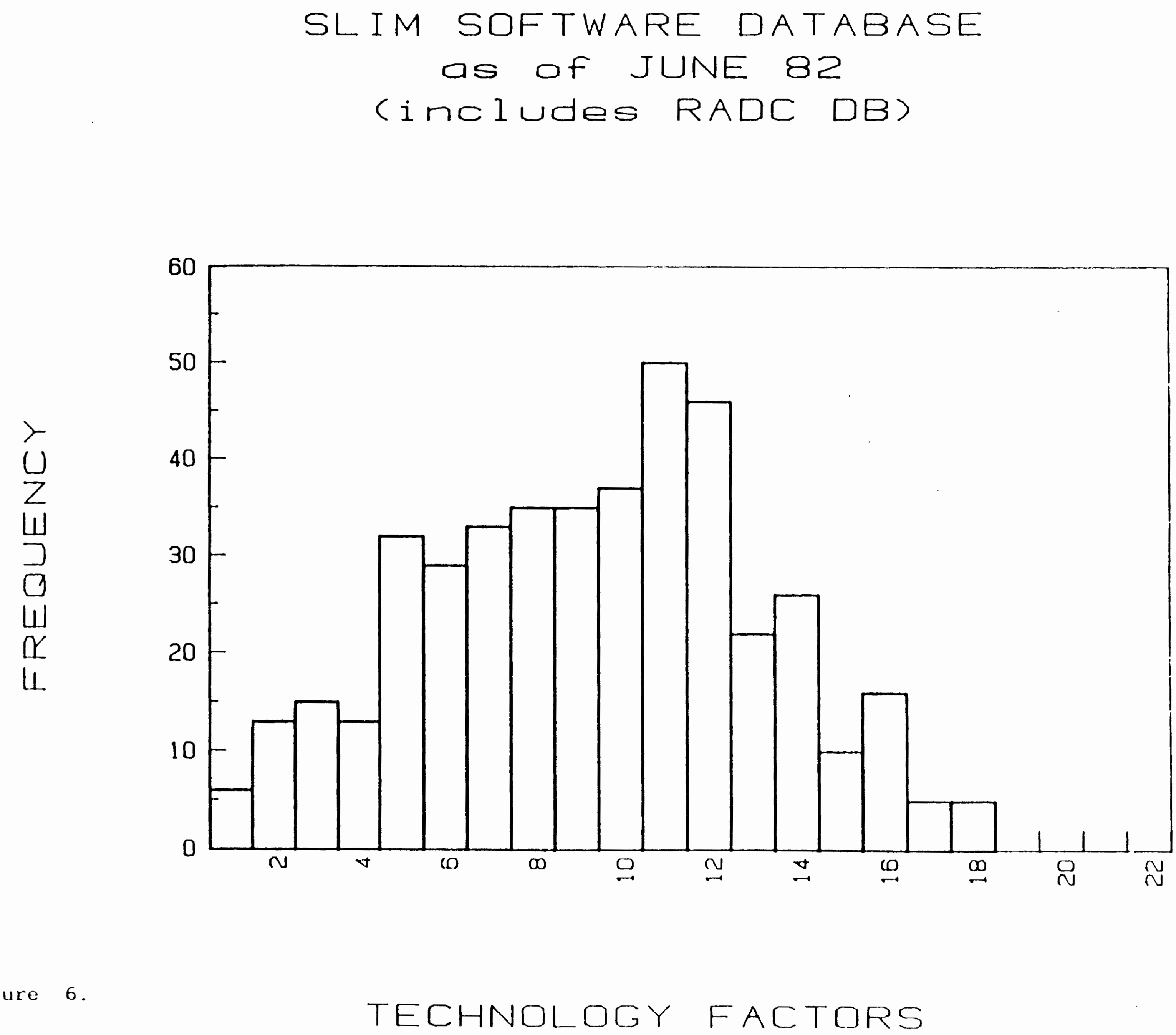

SLIM SOFTWARE DATABASE
as of JUNE 82
(includes RADC DB)
FREQUENCY
0
10
20
30
40
50
60
2
4
6
8
10
12
14
16
18
20
22
TECHNOLOGY FACTORS

Figure 6.

Table 1 shows the actual size, development time and development effort for each system. These numbers were run through the software equation to calculate the technology constant. Note that technology factor is used in the table. Technology factors are a linear sequence of number (1 - 22) that corresponds to a empirically determined sequence of C_k values from 610 - 121,393.

A parameter called the gradient level is also calculated. The gradient level is the ratio of life cycle effort divided by the development time cubed (K/t_d^3). Like C_k and the technology factor, a similar transformation of empirical values to a linear sequence is used, the range is from 1 - 6. The gradient level measures how fast you actually applied people to a project. For example, if a system's functional content is well known then one could build up to a high level of staffing quickly. Gradient levels 3, 4, 5 and 6 take on this general character in varying degrees. Gradient level 1 and 2 are very gradual manpower build ups. This profile seems most appropriate to match fuzzy, ill-defined and largely sequential problems. Time compression is not possible.

In certain situations where resources are constrained (people or money) it is possible to force a project to be built in a very sequential way even though high degrees of parallelism would have been possible. The gradient levels measure how projects were actually built. For example a Rebuild (Gradient Level 3) may have been built like a new complicated system (Gradient Level 1) because only 7 people were available to work on the system. These situations indicate that the software trade-off law is being exercised. Indeed four of the five systems here are cases where trade-offs did take place. These trade-off systems are the Exec System, Parts No., RFM and Materials.

Project Histories

Let's look at the technology factors calculated from the actual project data for each of these systems in the context of what we know about the nature of the work and the history of the project.

Cruise Missile

(Embedded System) - First an examination of the cruise missile project. The project data produced a technology factor of 1. This is the lowest value that has been observed historically.

This software was written to process a data stream of radar information related to the terrain on the ground below. The software was made up of 5800 machine order instructions. Limited memory conditions, maximum processing speed, and extremely complex algorithms make this software one of the most difficult to create. The very low technology factor in this case is determined by the extreme difficulty associated with this work. Micro-code and ROMable firmware tend to be on the extreme low end of the technology factor scale as well. These three problem classes appear to be equally difficult to solve.

EXEC System

(Systems Software) - The EXEC system can be classified as operating system software - a system executive. The EXEC system was designed to run on a standalone computer and handle all data and memory management functions. The system was written primarily in Fortran (80-85%) with the remaining (15-20%) being written in Assembly language for optimization purposes.

The EXEC system was done under contract by a large computer corporation. The contractor's project manager in charge of the software development was adamant about using a small team of people on the main software build (not more than 10-12 people). There was absolutely no way he was going to be talked into trying to 'steamroller' this project with large numbers of people. There were several reasons that he felt this way; the first reason was his own intuitive feel that small groups could communicate much better than large groups. From a management point of view he felt that a smaller team would be better able to implement the structured. modular development which he planned to use extensively on this system. The project manager used PDL, front end structured design as well as code walk throughs. This resulted in very modular and efficient code.

The system originally was designed and coded around a Interdata 7/32 CPU. This machine had some significant memory constraints which somewhat compromised the design. Approximately one third of the way through the development the software builders switched to a Perkin Elmer 3242 for a development machine. This CPU did not have the memory problems but the overall design remained consistent with its original concept. (It is not clear at this point whether the machine switch caused a break in the continuity of the project or not -- it usually does.) Both development computers supported on-line interactive development.

"EXEC" consisted of 61,000 lines of executable code. The development time from detailed logic design to full operational capability was 33 months and 248 manmonths of work was expended during the 33 month period.

(Verification of the Software Trade-Off Laws) - When these data were run through the software equation in SLIM it produced a Technology Factor of 8 (a technology constant of 4181) and a manpower acceleration rate (gradient level) of 1. It was described by the project manager as a gradient level 2 standalone system. This is often the clue to expect a trade-off situation because a level 1 system takes longer than a level 2 system to do. The gradient level calculation determines how it was actually done. So if the actual time was significantly longer than the minimum time for that type of system, it is very likely to be a trade-off candidate caused by some management constraint such as constrained funding, or constrained manpower. In this case a maximum of 10 to 12 people was the constraint. Using the actual size, the calibrated technology factor and the described level (61,000 Ss, TF = 8 and Level =2), SLIM's Monte Carlo simulation was run to determine the minimum time and corresponding effort. The simulation produced the following results:

Minimum Time	25.8 Month
Development Effort	658 Manmonths

This was encouraging since the actual time was 7.2 months longer than this, and the actual effort was significantly less.

We reasoned that if the parameters were set to 248 manmonths using the Design to Cost function in SLIM the software equation would calculate a time solution very close to the actual data point if the fourth power software trade-off law was valid.

This procedure resulted in a near perfect fit. A 29% stretch out in the schedule produced a 62% reduction in cost. Even if we had considered the system to be a level 1 system, the minimum time result would have been 28.5 months and 438 manmonths. The difference between this and the actual data point (33 months and 248 manmonths) is still significant -- an 18% stretch out produces a 48% reduction in cost. See Appendix A - a set of annotated outputs which show how the computerized model was used to verify the trade-off situation.

RFM, Parts No. and Materials

(Manufacturing Support Systems) - The final three data points are systems built by the same company. They are MIS systems that support manufacturing operations. The technology factors for these systems were very high (15,16 and 17). They fall in the top 10% of what has been observed in the U.S., Europe and Japan. The developing organization is very tool conscious. They upgrade and assimilate new equipment and software tools on a regular basis.

Over a four year period (1978 - 1982) this company moved out of a Non-IBM mainframe environment into a very modern large scale IBM environment. The software development people using the new equipment characterized the IBM environment as being much better for software development. The utilities that they formerly had to create were available as part of the normal IBM development tool kit. Additional capabilities available on this system included a data base management system and full capability screen editor. They were also using the ADE tools associated with IDMS. The functional designers made extensive use of the PRIDE structured development methodology. As a result they moved from a baseline technology factor of 12 in 1978 to 16 in 1982. This represents a phenominal increase in efficiency. Used in this way the technology factor is a very good measure of an organization's real efficiency increase.

Futhermore, this organization tends to invoke the software trade-off law strongly. They choose to build software using small groups of people (manpower constrained). Their experience shows that they get a higher quality product using this software development philosophy.

Materials System

(Verification of the Software Trade-Off Law) - It is worth taking a look at one of these systems - Materials. It is a very large system. Materials was built in the environment described above. The system contained 700,000 lines of COBOL. It took 38 months and 384 manmonths to build. The technology factor for Materials was 17. This is a very high technology factor (in the top 5% of what has been observed). The system designers described Materials as a rebuild of an existing system.

The actual data from Materials was run through SlIM in order to determine the minimum time to complete the project. Gradient level 3 was used because it corresponds to the fastest manpower buildup rate possible for a rebuilt system. The calibration of the actual data showed that they worked the problem like a level 1. This is often a good indication that a trade-off has taken place. Gradient level 1 has a more gradual manpower buildup therefore they take longer to do.

The Monte Carlo simulation routine in the SLIM model produced the following results:

Minimum time	27	Months
Development Effort	1434	Manmonths

To build the system in the minimum time it would have required 82 people at peak staffing. This organization typically doesn't use this many people on projects. Materials actually took 11 months longer to build than the minimum time but it used much less effort. To test the actual results against the 4th power software trade-off law the Design to Cost function in SLIM was used. If the actual recorded manmonths (384) was put in the Design to Cost function then the new schedule should be very close to 38 months. The software equation predicted a time of 37.45 months. The prediction was only off by 2 weeks from what actually happened. This is very close realizing that most people don't record the data more accurately than whole months.

The thirteen months schedule stretch out reduced the peak staffing to 16 people. The increase between the minimum time and the actual time is 39%. This time increase caused a 1,050 manmonth reduction in effort. The decrease between the minimum time effort and the actual effort is 73%. At a burdened labor rate of $50,000 / manyear this represents a 4.5 million dollar cost savings. See Appendix B - a set of annotated outputs showing how the SLIM computerized model can be used to perform this analysis.

(Materials System - Reliability) - The Materials system has some reliability information. SLIM shows that when Materials first became operational it had a Mean Time to Failure (MTTF) of about 1 week to 10 days. In other words the average time between major system failures that required prompt corrective action was about one week.

Some work in the software reliability modeling area gives us the capability to compare the expected reliability levels for the minimum time and the actual time for the Materials system and determine if there are any quality trade-offs between these different development approaches. When both cases were analyzed it appeared that there were significant differences. The minimum time solution would have produced roughly 1900 significant design and coding errors. The MTTF at the minimum time would have been about .05 months. This is about one eight hour day under normal operating conditions. On the other hand, the number of significant errors for the actual system would have been about 500 and the MTTF would have been .25 months (about 1 week). This is very close to the reliability level experienced in the field.

Why would the quality be better for the second case? In the manpower intensive software development people spend a great deal of time trying to communicate with one another. The human communication process is ambiguous. Therefore erroneous human communication is always happening. Software developments that have large amounts of human communication tend to generate more "noise". This means that such systems have a large number of errors and a short mean time between failures. By reducing the human communication on a project you also reduce the noise. A few good people given enough time will not create nearly as many design or coding errors. The attached computerized output shows how the quality changes between the two solutions.

!!!

RELIABILITY

!!!

MATERIALS SYSTEM

MINIMUM TIME RELIABILITY PARAMETERS

A SUMMARY OF THE CURRENT PARAMETERS ARE:

TIME:	27.1 MONTHS	The minimum time solution
EFFORT:	1434 MANMONTHS	
COST:	5927 (X 1000 $)	
MEAN TIME TO FAILURE:	.05 MONTHS ←	1 eight hour day of normal operation
EXPECTED ERRORS:	1871 ERRORS ←	Close to 1900 significant design and coding errors
EXPECTED ERRORS/1000 SS:	2.67 ERRORS	
EXPECTED ERRORS/1000 SS (FROM SIT TO FOC):	.47 ERRORS	
ERRORS REMAINING AT 27.1 Mos:	93 ERRORS	

!!!

RELIABILITY

!!!

MATERIALS SYSTEM

ACTUAL DEVELOPMENT SCHEDULE RELIABILITY PARAMETERS

A SUMMARY OF THE CURRENT PARAMETERS ARE:

TIME:	37.5 MONTHS	
EFFORT:	384 MANMONTHS	
COST:	1600 (X 1000 $)	
MEAN TIME TO FAILURE:	.25 MONTHS ←	1 week between major system failures
EXPECTED ERRORS:	501 ERRORS ←	Only about 500 significant errors
EXPECTED ERRORS/1000 SS:	.72 ERRORS	
EXPECTED ERRORS/1000 SS (FROM SIT TO FOC):	.13 ERRORS	
ERRORS REMAINING AT 37.5 Mos:	25 ERRORS	

THE TABLE BELOW SHOWS THE EXPECTED ERROR RATE, MEAN TIME TO FAILURE (MONTHS), AND EXPECTED CUMULATIVE ERRORS FOR DEVELOPMENT THROUGH THE .999 RELIABILITY LEVEL. THESE VALUES ARE BASED ON A DEVELOPMENT TIME OF 27.1 MONTHS AND A TOTAL DEVELOPMENT EFFORT OF 1434.2 MANMONTHS.

MINIMUM TIME ERROR FORCAST

EXPECTED ERRORS

MONTH	MEAN ERROR RATE	ERROR RATE RANGE	EXPECTED CUM ERRORS FIXED	RANGE CUM ERRORS FIXED	MTTF (MONTHS)
JAN 79	15.2	10.7 - 19.7	8	6 - 9	-
FEB 79	30.1	21.4 - 38.8	30	25 - 35	-
MAR 79	44.2	31.7 - 56.7	68	56 - 79	-
APR 79	57.3	41.4 - 73.2	118	99 - 138	-
MAY 79	69.0	50.2 - 87.8	182	152 - 212	-
JUN 79	79.2	58.0 - 100.3	256	214 - 298	-
JUL 79	87.6	64.6 - 110.6	339	284 - 395	-
AUG 79	94.2	69.9 - 118.4	430	360 - 501	-
SEP 79	98.8	73.8 - 123.8	527	440 - 614	-
OCT 79	101.6	76.4 - 126.8	628	524 - 731	-
NOV 79	102.6	77.6 - 127.6	730	610 - 850	-
DEC 79	101.9	77.4 - 126.3	832	695 - 969	-
JAN 80	99.6	76.1 - 123.1	933	779 - 1087	-
FEB 80	96.1	73.8 - 118.4	1031	861 - 1201	-
MAR 80	91.5	70.6 - 112.3	1125	940 - 1310	-
APR 80	86.0	66.6 - 105.3	1214	1014 - 1413	-
MAY 80	79.8	62.2 - 97.5	1296	1083 - 1510	-
JUN 80	73.2	57.3 - 89.2	1373	1147 - 1599	-
JUL 80	66.5	52.2 - 80.7	1443	1205 - 1680	.02
AUG 80	59.7	47.0 - 72.3	1506	1258 - 1754	.02
SEP 80	53.0	41.9 - 64.1	1562	1305 - 1819	.02
OCT 80	46.6	36.9 - 56.2	1612	1347 - 1877	.02
NOV 80	40.5	32.2 - 48.8	1656	1383 - 1928	.02
DEC 80	34.9	27.8 - 41.9	1693	1414 - 1972	.03
JAN 81	29.7	23.8 - 35.7	1725	1441 - 2010	.03
FEB 81	25.1	20.1 - 30.1	1753	1464 - 2041	.04
MAR 81	21.0	16.9 - 25.1	1776	1483 - 2068	.05
APR 81	17.4	14.0 - 20.8	1795	1499 - 2091	.06
MAY 81	14.3	11.5 - 17.0	1811	1513 - 2109	.07
JUN 81	11.6	9.4 - 13.8	1824	1523 - 2124	.09
JUL 81	9.3	7.6 - 11.1	1834	1532 - 2136	.11
AUG 81	7.5	6.0 - 8.9	1842	1539 - 2146	.13
SEP 81	5.9	4.8 - 7.0	1849	1545 - 2154	.17
OCT 81	4.6	3.8 - 5.5	1854	1549 - 2160	.22
NOV 81	3.6	2.9 - 4.3	1858	1552 - 2164	.28
DEC 81	2.8	2.2 - 3.3	1862	1555 - 2168	.36
JAN 82	2.1	1.7 - 2.5	1864	1557 - 2171	.47
FEB 82	1.6	1.3 - 1.9	1866	1559 - 2173	.63
MAR 82	1.2	1.0 - 1.4	1867	1560 - 2175	.84
APR 82	.9	.7 - 1.1	1868	1561 - 2176	1.13
MAY 82	.7	.5 - .8	1869	1561 - 2177	1.53
JUN 82	.5	.4 - .6	1870	1562 - 2177	2.10

No MTTF because we don't have a system yet

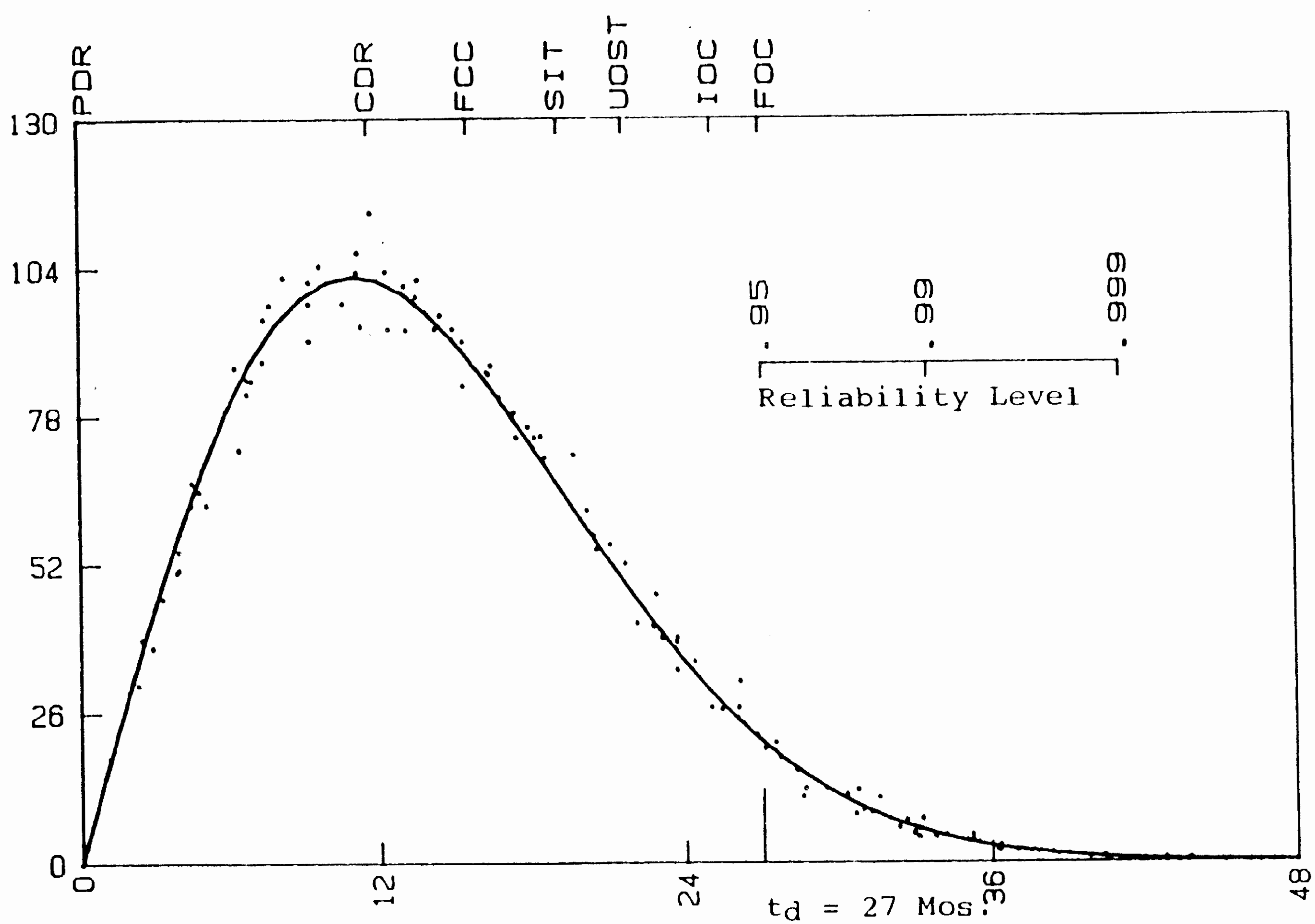
Systems Integration Test starts about here.

MTTF = .05 Mos. when the system will become operati

You will have to continue to te and work an additional 7.5 mont to get a MTTF of 1 week. The development cost now will be close to 10 million dollars.

EXPECTED ERROR RATE

Minimum Time Case



DEV TIME (months)

MATERIALS SYSTEM
SS 700000
TF 17
LEVEL 3

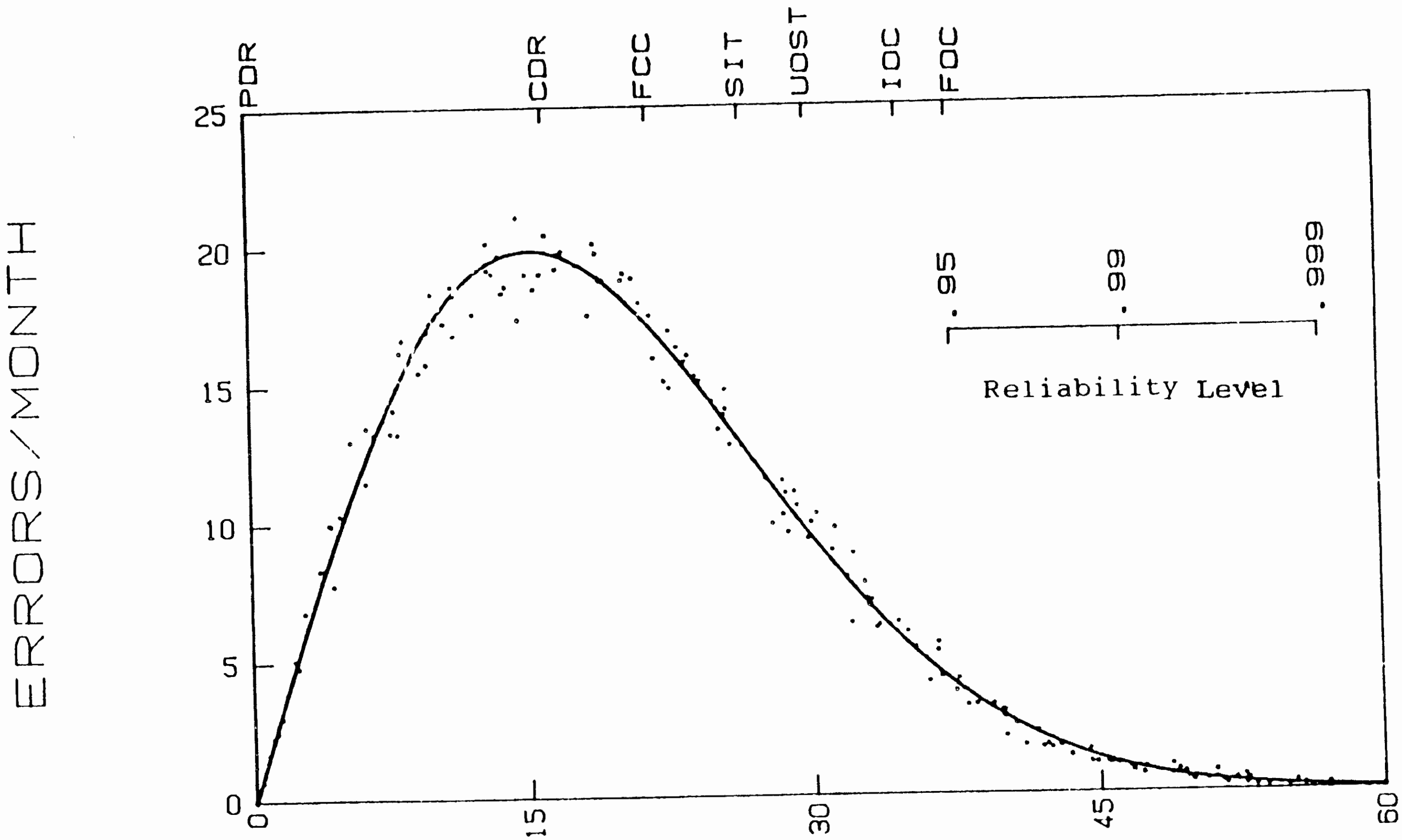
EXPECTED ERRORS

THE ACTUAL SCHEDULE ERROR FORCAST

MONTH	MEAN ERROR RATE	ERROR RATE RANGE	EXPECTED CUM ERRORS FIXED	RANGE CUM ERRORS FIXED	MTTF (MONTHS)	
JAN 79	2.1	1.4 - 2.9	1	1 - 1	-	
FEB 79	4.2	2.8 - 5.7	4	4 - 5	-	
MAR 79	6.3	4.2 - 8.4	10	8 - 11	-	
APR 79	8.3	5.6 - 11.0	17	14 - 20	-	
MAY 79	10.2	6.9 - 13.4	26	22 - 31	-	
JUN 79	11.9	8.2 - 15.6	37	31 - 44	-	
JUL 79	13.5	9.3 - 17.7	50	41 - 58	-	
AUG 79	14.9	10.4 - 19.5	64	53 - 75	-	
SEP 79	16.2	11.3 - 21.1	80	66 - 93	-	
OCT 79	17.3	12.2 - 22.4	96	80 - 113	-	
NOV 79	18.2	12.9 - 23.5	114	95 - 134	-	
DEC 79	18.9	13.4 - 24.4	133	110 - 156	-	No system yet
JAN 80	19.4	13.9 - 24.9	152	126 - 178	-	
FEB 80	19.7	14.2 - 25.3	172	142 - 201	-	
MAR 80	19.9	14.3 - 25.4	191	158 - 224	-	
APR 80	19.8	14.4 - 25.3	211	175 - 247	-	
MAY 80	19.6	14.3 - 25.0	231	191 - 271	-	
JUN 80	19.3	14.1 - 24.5	250	207 - 293	-	
JUL 80	18.8	13.8 - 23.8	269	223 - 316	-	
AUG 80	18.2	13.4 - 23.0	288	239 - 337	-	
SEP 80	17.5	13.0 - 22.1	306	253 - 358	-	
OCT 80	16.7	12.5 - 21.0	323	268 - 379	-	
NOV 80	15.9	11.9 - 19.9	339	281 - 398	-	
DEC 80	15.0	11.2 - 18.8	355	294 - 416	-	
JAN 81	14.1	10.6 - 17.6	369	306 - 433	-	
FEB 81	13.1	9.9 - 16.3	383	317 - 449	.08	Systems Integration Test
MAR 81	12.2	9.2 - 15.1	396	328 - 464	.08	
APR 81	11.2	8.5 - 13.9	407	337 - 477	.09	
MAY 81	10.3	7.8 - 12.7	418	346 - 490	.10	
JUN 81	9.4	7.2 - 11.6	428	354 - 501	.11	
JUL 81	8.5	6.5 - 10.5	437	362 - 512	.12	
AUG 81	7.7	5.9 - 9.5	445	368 - 521	.13	
SEP 81	6.9	5.3 - 8.5	452	374 - 530	.15	
OCT 81	6.1	4.7 - 7.6	459	380 - 537	.16	
NOV 81	5.5	4.2 - 6.7	464	385 - 544	.18	
DEC 81	4.8	3.7 - 5.9	470	389 - 550	.21	
JAN 82	4.2	3.3 - 5.2	474	393 - 556	.24	
FEB 82	3.7	2.9 - 4.5	478	396 - 560	.27	MTTF = 1 week at full operational capability
MAR 82	3.2	2.5 - 3.9	482	399 - 564	.31	
APR 82	2.8	2.2 - 3.4	485	401 - 568	.36	
MAY 82	2.4	1.9 - 2.9	487	404 - 571	.41	
JUN 82	2.1	1.6 - 2.5	489	405 - 574	.48	
JUL 82	1.8	1.4 - 2.1	491	407 - 576	.57	
AUG 82	1.5	1.2 - 1.8	493	408 - 578	.67	How things get better as you continue to test and work.
SEP 82	1.3	1.0 - 1.5	494	409 - 579	.79	
OCT 82	1.1	.8 - 1.3	496	410 - 581	.94	
NOV 82	.9	.7 - 1.1	497	411 - 582	1.12	
DEC 82	.7	.6 - .9	497	412 - 583	1.34	
JAN 83	.6	.5 - .7	498	412 - 584	1.62	
FEB 83	.5	.4 - .6	499	413 - 584	1.96	
MAR 83	.4	.3 - .5	499	413 - 585	2.38	
APR 83	.3	.3 - .4	499	414 - 585	2.91	
MAY 83	.3	.2 - .3	500	414 - 586	3.58	
JUN 83	.2	.2 - .3	500	414 - 586	4.42	
JUL 83	.2	.1 - .2	500	414 - 586	5.47	
AUG 83	.1	.1 - .2	500	414 - 586	6.82	

EXPECTED ERROR RATE

For Trade-off Case



t_d = 37.5 Mos.

DEV TIME (months)

MATERIALS SYSTEM
SS 700000
TF 17
LEVEL 3

Plotting the Data on the RADC Trend Lines

We have examined three classes of software; a realtime embedded system, an operating system and three manufacturing support MIS systems. The different schedule. staffing and quality implications that determined the particular software development strategy have also been examined. Now we will plot these data on the trend lines established by the RADC analysis. The data will be portrayed on the graphs listed below:

1. Productivity versus System Size
2. Project Duration versus System Size
3. Total Manmonths versus System Size
4. Average Manpower versus System Size

See Figures 7 through 10 .

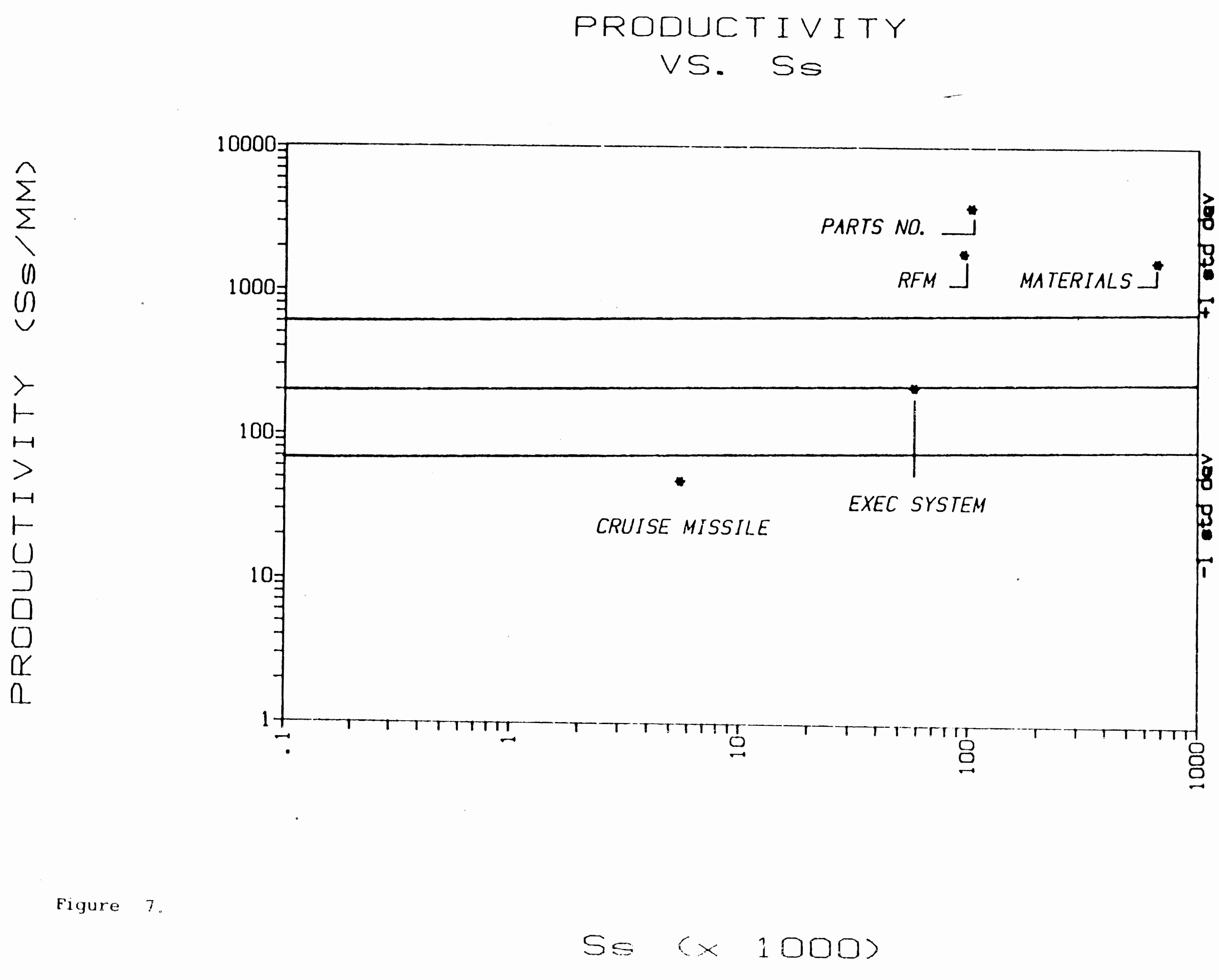

PRODUCTIVITY
VS. Ss
PRODUCTIVITY (Ss/MM)
10000
1000
100
10
1
.1
1
10
100
1000
PARTS NO.
RFM
MATERIALS
EXEC SYSTEM
CRUISE MISSILE
+1 std dev
-1 std dev
Ss (x 1000)

Figure 7.

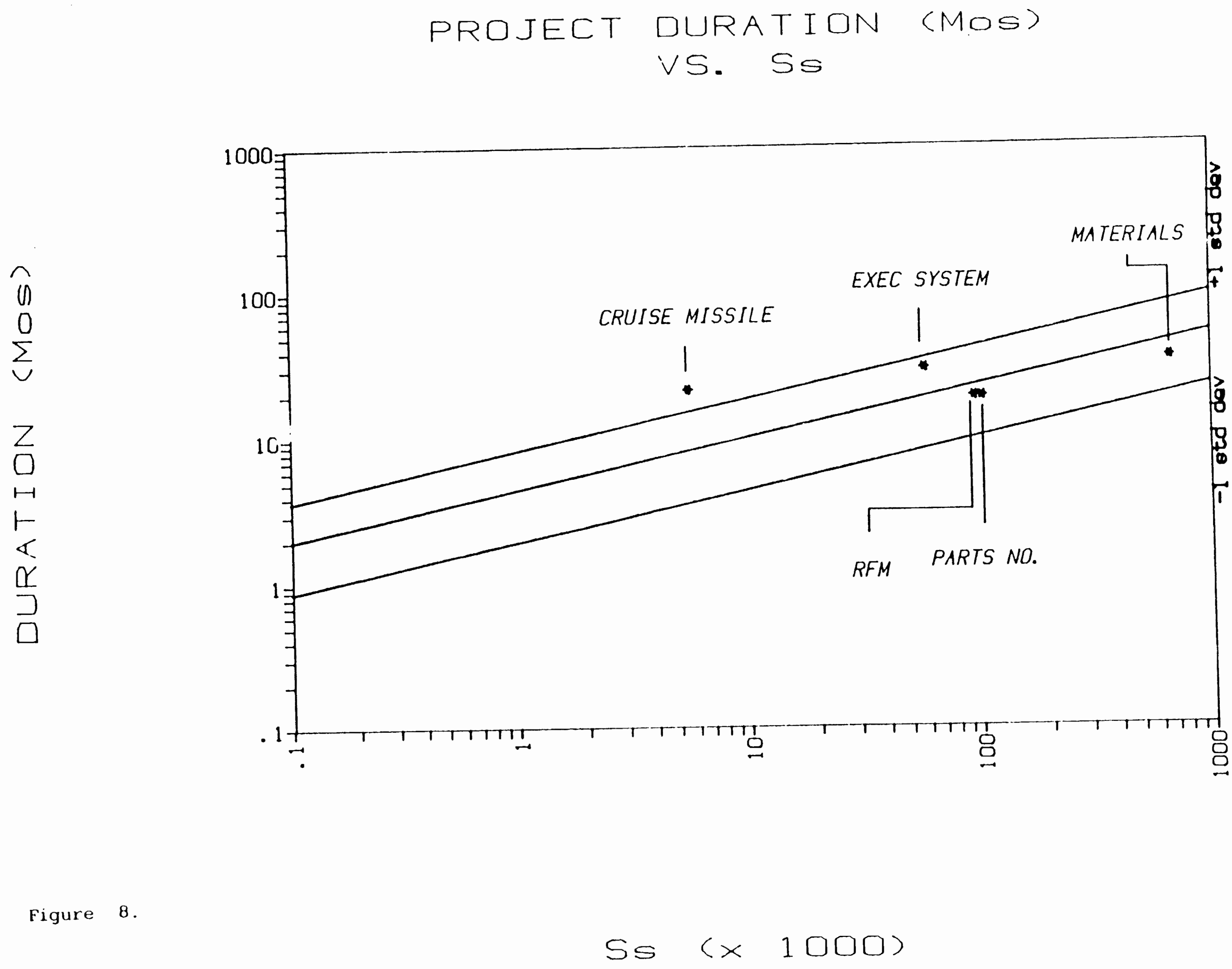

PROJECT DURATION (Mos)
VS. Ss
+1 std dev
-1 std dev
MATERIALS
EXEC SYSTEM
CRUISE MISSILE
RFM
PARTS NO.
1000
100
10
1
.1
DURATION (Mos)
Ss (x 1000)

Figure 8.

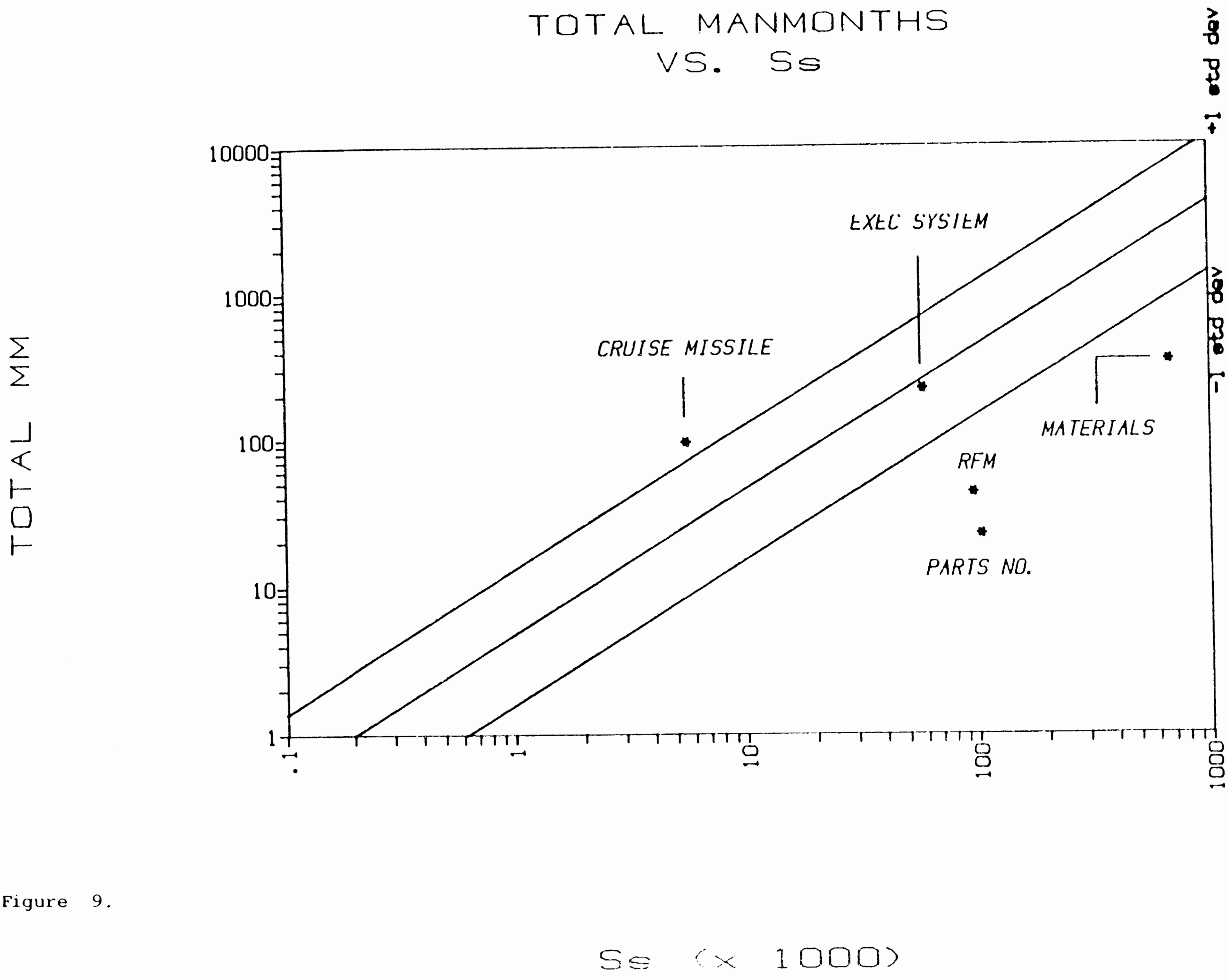

TOTAL MANMONTHS
VS. Ss
TOTAL MM
Ss (x 1000)
10000
1000
100
10
1
.1
1
10
100
1000
+1 std dev
-1 std dev
EXEC SYSTEM
CRUISE MISSILE
MATERIALS
RFM
PARTS NO.

Figure 9.

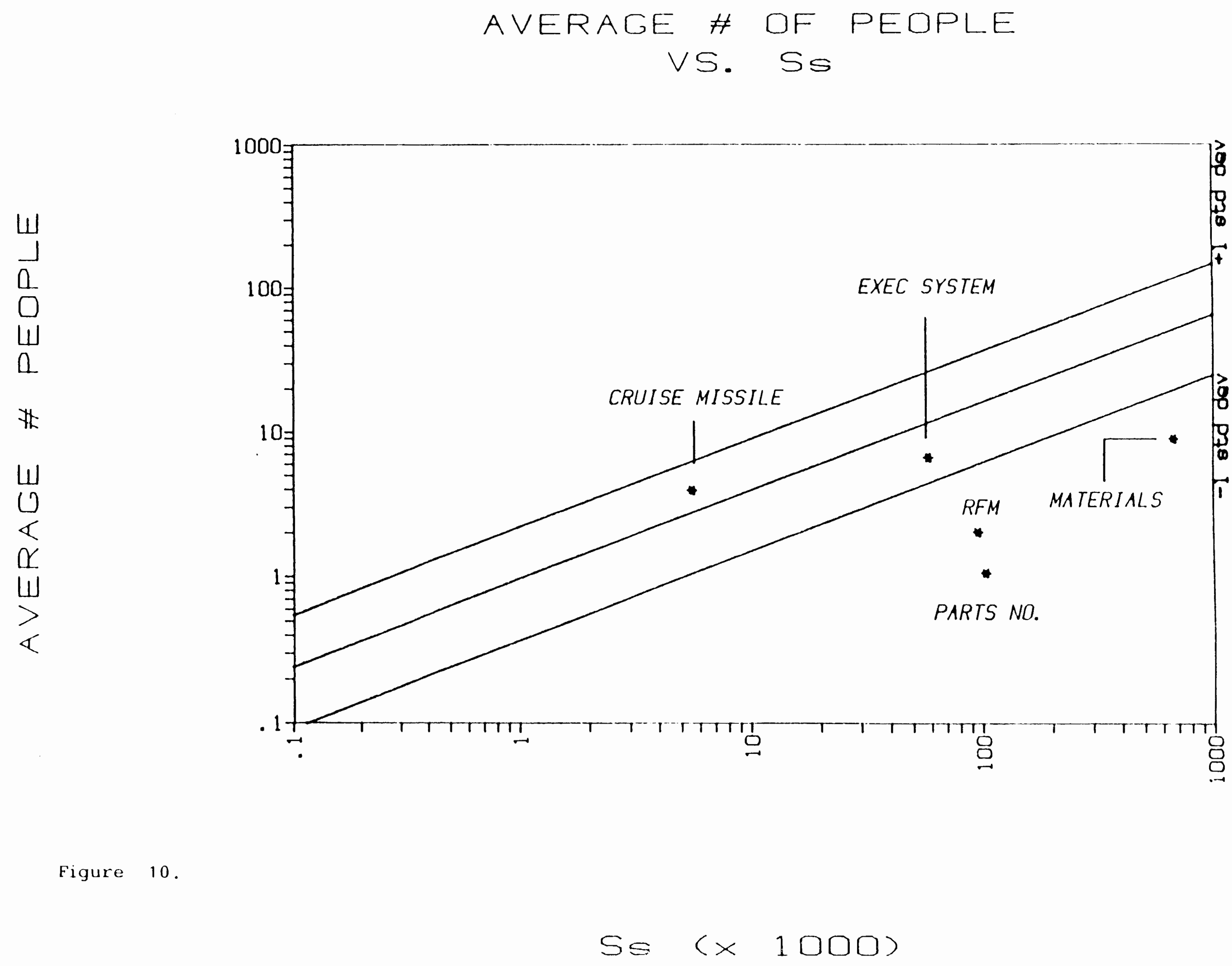

AVERAGE # OF PEOPLE
VS. Ss
AVERAGE # PEOPLE
1000
100
10
1
.1
Ss (x 1000)
+1 std dev
-1 std dev
CRUISE MISSILE
EXEC SYSTEM
RFM
PARTS NO.
MATERIALS

Figure 10.

Productivity Versus System Size

The first graph that will be analyzed (Figure 7.) is Productivity versus System Size. The cruise missile software falls more than one standard deviation below the average productivity of over 400 systems. Remember this software was extremely complex. it had a very low technology factor (1 on a scale of 1 - 22) and it was a minimum time (maximum effort) development strategy. The cruise missile's productivity plots exactly where one would expect (very low).

The EXEC system's productivity plots almost exactly on the average trend line. The technology factor for this system was 8. This is lower than the average technology factor of all the systems in the RADC data base. Normally, with this technology factor we would expect to see a slightly lower than average productivity. However this would only be true if a minimum schedule (maximum effort) strategy were employed. Productivity for the minimum time would be 61,000/665MM or 93 source statements per manmonth of work. This productivity would plot well below the average trend line. We know that this system was resource constrained at a maximum staffing of 11 people. The manpower constraint caused a 7.2 month increase in the schedule and a substantial reduction in effort. The calculated productivity for the system as it was actually built is 61.000/248MM or 246 source statements per manmonth of work. This productivity is much higher than the minimum time. This is why the EXEC system's productivity plots on the average trend line even though a technology factor of 8 might initially cause us to think it would be lower.

The three manufacturing support systems experienced very high productivity rates. All three of these systems had very high technology factors. They were all resource constrained by the organization's small team development approach. Consequently, all three of these systems fall in the trade-off region. This organization's capital investment in tools coupled with their development philosophy has really paid off. Their productivity plots 2 to 3 standard deviations higher than the average of over 400 systems.

Project Duration Versus System Size

The second graph is project Duration versus System Size. Notice first that the cruise missile software plots more than one standard deviation longer than the average of 400 systems. The very complex nature of this work would intuitively lead you to expect this and indeed, this is the case.

The EXEC system plots close to one standard deviation longer than the average duration. Why? Because the schedule was deliberately stretched out 7.2 months from the minimum time schedule - - the trade-off becomes very evident.

The three manufacturing support systems all plot very close to the average duration. The influence of the high technology factors shortens the development schedule at the minimum time. Materials schedule was stretched out 13 months. Even with this dramatic time stretch out it still plots slightly shorter than the average duration. The other two systems had similar situations.

Total Manmonths Versus System Size

The next graph is Total Manmonths versus System Size (Figure 9). Manmonths are proportional to cost therefore this graph also represents how expensive it is to build a system. Notice that the three manufacturing systems required much less effort than the average (2 to 3 standard deviations fewer manmonths). This means that these systems were very inexpensive compared to what other organizations have historically paid for similar sized systems. Again, it shows quite clearly that this organization's capital investments and development practices have really paid off.

The EXEC system plots slightly less than the average manmonths. This is due to the constrained resource development approach (Trade-Off) that was used.

The cruise missile was more complex. It had a lower productivity, it took longer to build and it required more effort. The effort was more than one standard deviation (higher) than the average of over 400 systems.

Average Manpower Versus System Size

The last graph is Average Manpower versus System Size (Figure 10). As you might expect, the cruise missile software required a greater than average number of people.

The EXEC system took less than the average number of people. This is not surprising because the system was resource constrained at 11 people.

The manufacturing support systems used significantly fewer people. All three systems fall two to three standard deviations lower than the average number of people of 400 systems.

CONCLUSION

The graphs tend to support that the software equation is very close to expressing the way software systems behave with respect to changes in the schedule and effort. The large number of systems in the RADC database from which the trend lines on the four graphs were derived provide an objective means to measure the actual performance. The technology factor provides the capability to measure relative increases in efficiency in a homogenous class of work.

When these metrics are combined they give a good account of how you really performed. This starts to become meaningful as you acquire tools and develop new methodologies. Measure the old projects. Measure the new projects. Did you really get the pay off that you anticipated? What is the pay off? Are you getting an equal or better quality product in less time, using less effort and fewer people?

Using this approach we are able to capture the long term dynamics of the software development process. As things change over time we will be able to better assess the actual capabilities of the organization. This information can then be fed back into the estimating process so that our estimates are consistent with the organization's current capabilities.

APPENDIX A

EXEC SYSTEM
VERIFICATION OF THE TRADE-OFF LAW

```
******************************************************************************
                       SUMMARY OF INPUT PARAMETERS
******************************************************************************
SYSTEM:  EXEC SYSTEM                                        DATE:  10 MAR 83
                                                            TIME:      13:30

PROJECT START:  0678

COST ELEMENTS
  COST/MY:                75000.            INFLATION RATE:        .100
  MONETARY UNIT:    ($)
  STD DEV (COST/MY):       7500.

ENVIRONMENT
  ONLINE DEVELOPMENT:       1.00                HOL USAGE:          .85
  DEVELOPMENT TIME:         1.00          PRODUCTION TIME:         0.00
  DBMS:                     0.00            REPORT WRITER:         0.00
  LANGUAGE:              FORTRAN

SYSTEM
  TYPE:  OPERATING SYSTEM                  REAL-TIME CODE:          .15
  LEVEL:                       2  <----       UTILIZATION:          .80

MODERN PROGRAMMING PRACTICES
  STRUCTURED PROGRAMMING:  > 75%      DESIGN/CODE INSPECTION:     > 75%
  TOP-DOWN DESIGN:         > 75%      CHIEF PROG TEAM USAGE:      < 25%

EXPERIENCE
  OVERALL:             EXTENSIVE               SYSTEM TYPE:  EXTENSIVE
  LANGUAGE:              AVERAGE                  HARDWARE:    MINIMAL

TECHNOLOGY
  FACTOR:                      8  <----
SIZE
  LOWEST:                  51000                   HIGHEST:      71000
------------------------------------------------------------------------------
```

$C_k = 4181$

Gradient level 2 for a standalone system as described by the project manager.

Technology factor 8 was calculated from the actua project data.

!!

SIMULATION

!!

EXEC SYSTEM
10 MAR 83
13:36

	MEAN	STD DEV
SYSTEM SIZE (STATEMENTS)	61000	3333
MINIMUM DEVELOPMENT TIME (MONTHS)	25.8	.8
DEVELOPMENT EFFORT (MANMONTHS)	658.1	70.6
DEVELOPMENT COST ($)		
(UNINFLATED)	4146	584
(INFLATED)	4599	655
PEAK MANPOWER (PEOPLE)	38	

This is the fastest time that the EXEC system could have been built in.

To build the system in the minimum time it would have required 38 people at peak staffing.

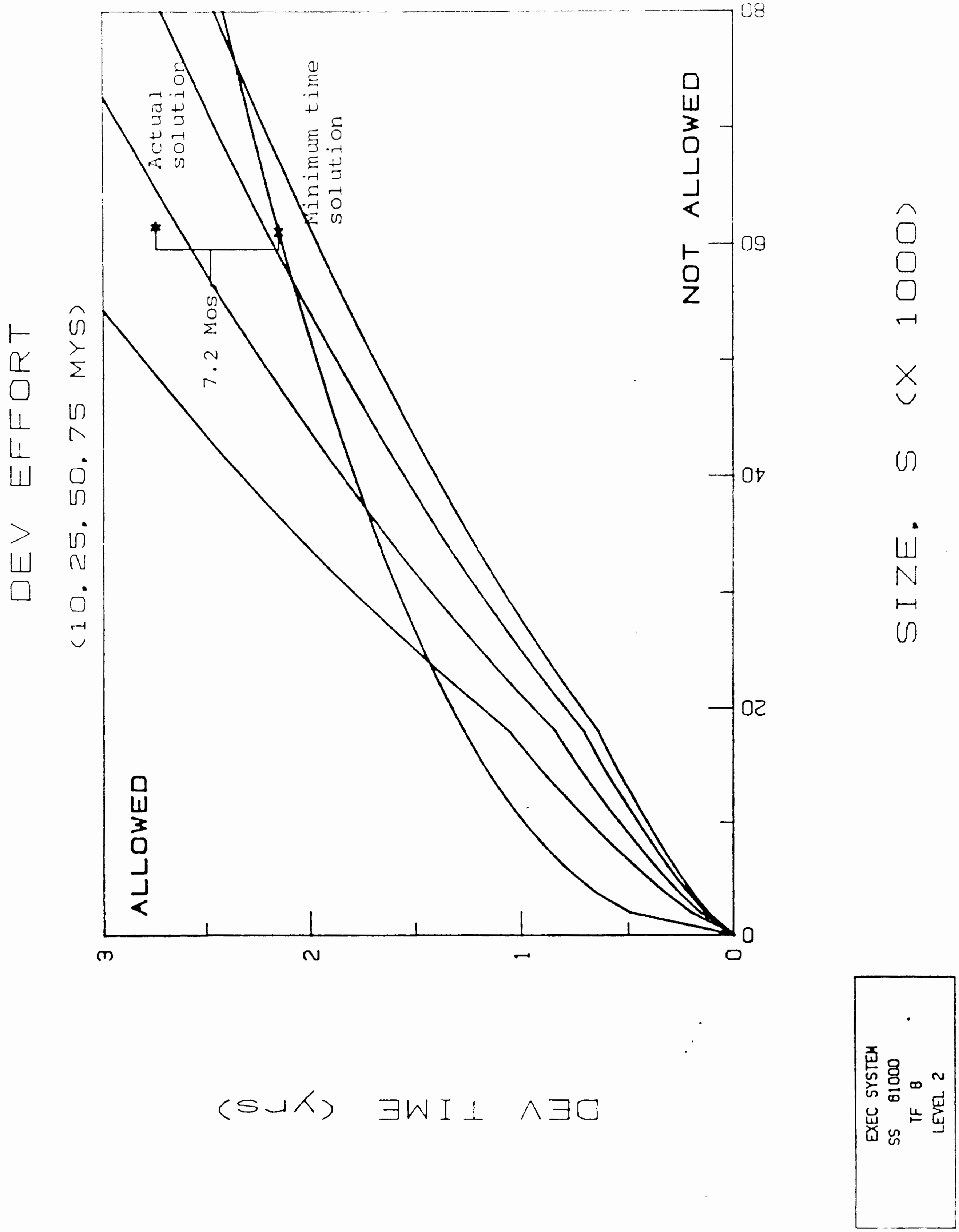
SENSITIVITY PROFILE FOR MINIMUM TIME SOLUTION

	SOURCE STMTS	MONTHS	MANMONTHS	UNINFLATED COST (X 1000)
-3 STD DEV	51000.	23.9	510.	3187.
-1 STD DEV	57667.	25.2	614.	3835.
MOST LIKELY	61000.	25.8	658.	4146.
+1 STD DEV	64333.	26.4	718.	4490.
+3 STD DEV	71000.	27.5	824.	5152.

A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

TOTAL MANMONTHS	658	IN NORMAL RANGE
PROJECT DURATION (MONTHS)	25.8	IN NORMAL RANGE
AVERAGE # PEOPLE	25	IN NORMAL RANGE
PRODUCTIVITY (LINES/MM)	93	IN NORMAL RANGE

Productivity at the minimum time is 93 Ss per manmonth.


DEV EFFORT
(10, 25, 50, 75 MYS)
ALLOWED
NOT ALLOWED
Actual solution
Minimum time solution
7.2 Mos
0
1
2
3
0
20
40
60
80
DEV TIME (yrs)
SIZE, S (X 1000)
EXEC SYSTEM
SS 81000
TF 8
LEVEL 2

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                          DESIGN TO COST
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                           EXEC SYSTEM


THE BEST ESTIMATES OF THE MINIMUM TIME AND CORRESPONDING EFFORT AND COST TO
DEVELOP YOUR SYSTEM ARE:

            MINIMUM TIME:          25.85 MONTHS
            EFFORT:                  658 MANMONTHS
            COST:                   4146 (X 1000 $)

ENTER DESIRED DEVELOPMENT EFFORT IN MANMONTHS
?
248
```

← The actual manmonths of effort

```
                        NEW DEVELOPMENT TIME
                           EXEC SYSTEM

                                        MEAN    STD DEV
                                        ----    -------

       NEW DEV TIME (MONTHS)            33.00       .98
       NEW DEV EFFORT (MANMONTHS)        248         27
       NEW DEV COST (X 1000)            1550        218
```

A near perfect fit to the inverse 4th trade-off relationship.

```
YOUR FILE IS NOW UPDATED WITH THESE NEW PARAMETERS.  RUN MANLOADING AND CASHFLOW
TO SEE HOW THESE SAVINGS CAN BE REALIZED.


A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:


  TOTAL MANMONTHS              248         IN NORMAL RANGE
  PROJECT DURATION (MONTHS)   33.0         IN NORMAL RANGE
  AVERAGE # PEOPLE               8         IN NORMAL RANGE
  PRODUCTIVITY (LINES/MM)      246         IN NORMAL RANGE
```

← Productivity has increased from 93/MM to 246/MM..

!!

MANLOADING

!!

EXEC SYSTEM

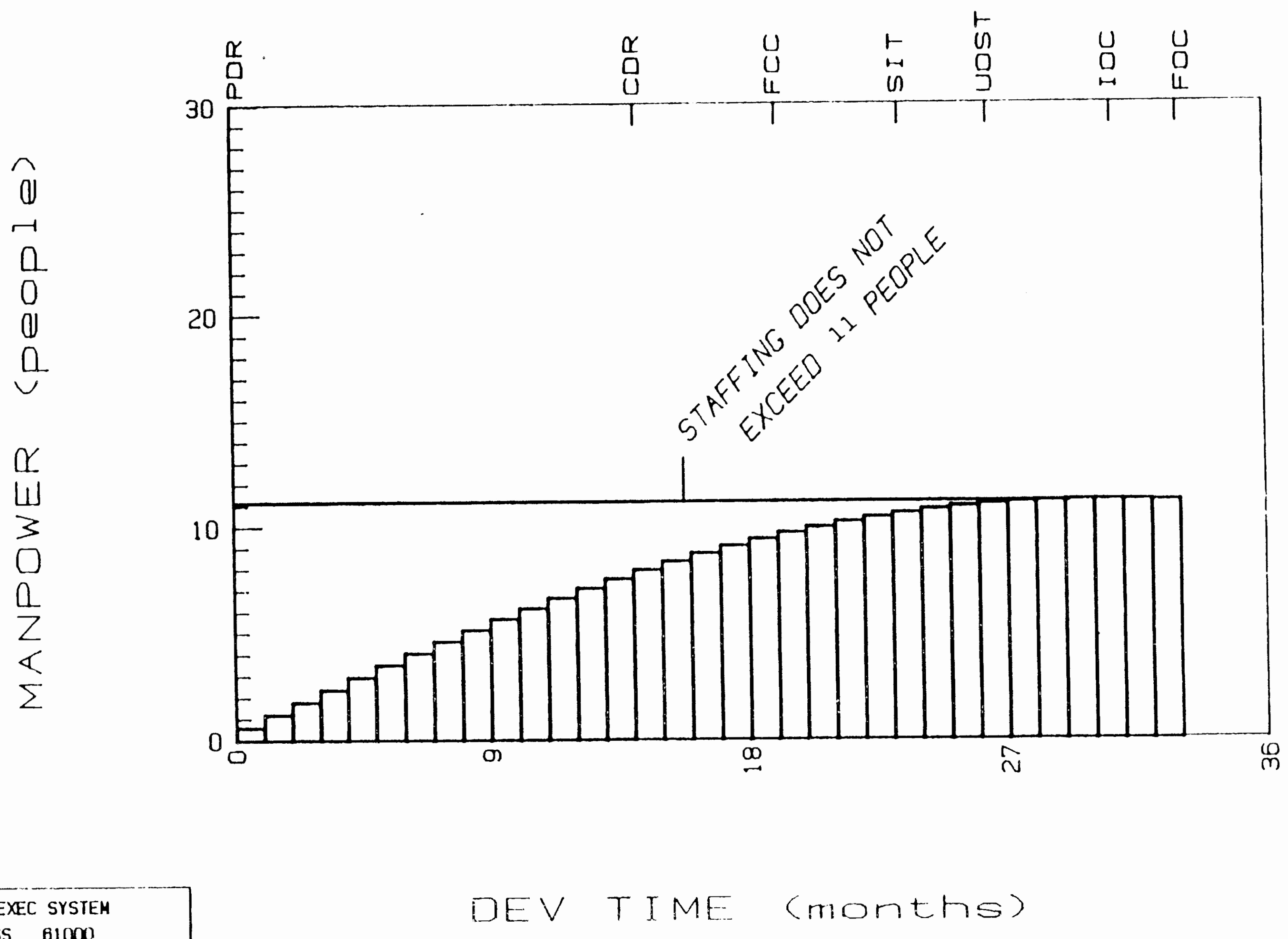
THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT (AND STANDARD DEVIATION) REQUIRED FOR DEVELOPMENT. THESE VALUES ARE BASED ON A DEVELOPMENT TIME OF 33.0 MONTHS AND A TOTAL DEVELOPMENT EFFORT OF 248.0 MANMONTHS.

STAFFING PLAN

MONTH	PEOPLE/MONTH	STD DEV	CUMULATIVE MANMONTHS	CUM STD DEV
JUN 78		.1		
JUL 78	1	.2	1	
AUG 78	1	.3	3	
SEP 78	2	.4	5	1
OCT 78	3	.5	7	1
NOV 78	3	.6	11	1
DEC 78	4	.7	14	2
JAN 79	4	.8	19	2
FEB 79	5	.9	24	3
MAR 79	5	.9	29	3
APR 79	6	1.0	35	4
MAY 79	6	1.1	41	4
JUN 79	7	1.1	48	5
JUL 79	7	1.2	56	6
AUG 79	8	1.2	63	7
SEP 79	8	1.3	72	8
OCT 79	9	1.3	80	9
NOV 79	9	1.4	89	10
DEC 79	9	1.4	98	11
JAN 80	10	1.4	108	12
FEB 80	10	1.5	118	13
MAR 80	10	1.5	128	14
APR 80	10	1.5	138	15
MAY 80	11	1.5	149	16
JUN 80	11	1.5	159	17
JUL 80	11	1.5	170	18
AUG 80	11	1.5	181	19
SEP 80	11	1.5	192	21
OCT 80	11	1.5	203	22
NOV 80	11	1.5	214	23
DEC 80	11	1.5	226	24
JAN 81	11	1.5	237	25
FEB 81	11	1.5	248	27
MAR 81	6	.7	254	28

Peak staffing does not exceed 11 people (this is the maximum number of people the project manager was willing to use).

STAFFING PLAN



EXEC SYSTEM
SS 61000
TF 8

APPENDIX B

MATERIALS SYSTEM
VERIFICATION OF THE TRADE-OFF LAW

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                        SUMMARY OF INPUT PARAMETERS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                                        DATE:  10 MAR 83
SYSTEM:  MATERIALS SYSTEM                               TIME:     12:26

PROJECT START:  0179

COST ELEMENTS
  COST/MY:                 50000.            INFLATION RATE:      .100
  MONETARY UNIT:   ($)
  STD DEV (COST/MY):        5000.

ENVIRONMENT
  ONLINE DEVELOPMENT:       1.00                  HOL USAGE:      1.00
  DEVELOPMENT TIME:         1.00            PRODUCTION TIME:      0.00
  DBMS:                      .30              REPORT WRITER:      0.00
  LANGUAGE:                COBOL

SYSTEM
  TYPE:  BUSINESS APPLICATION                REAL-TIME CODE:      0.00
  LEVEL:                       3 <---           UTILIZATION:       .50

MODERN PROGRAMMING PRACTICES
  STRUCTURED PROGRAMMING:  > 75%      DESIGN/CODE INSPECTION:     > 75%
  TOP-DOWN DESIGN:         > 75%      CHIEF PROG TEAM USAGE:      < 25%

EXPERIENCE
  OVERALL:               EXTENSIVE               SYSTEM TYPE:   AVERAGE
  LANGUAGE:              EXTENSIVE                  HARDWARE:  EXTENSIVE

TECHNOLOGY
  FACTOR:                     17 <---

SIZE
  LOWEST:                600000                      HIGHEST:    800000
--------------------------------------------------------------------------
```

A rebuild of an existin[g] system.

$C_k = 35422$

Exceptionally high.
A very good performer.

!!!

SIMULATION

!!!

MATERIALS SYSTEM
10 MAR 83
12:27

	MEAN	STD DEV
SYSTEM SIZE (STATEMENTS)	700000	33333
MINIMUM DEVELOPMENT TIME (MONTHS)	27.1	.7
DEVELOPMENT EFFORT (MANMONTHS)	1434.2	143.1
DEVELOPMENT COST ($)		
(UNINFLATED)	5927	812
(INFLATED)	6606	898
PEAK MANPOWER (PEOPLE)	82	

This is the absolute minimum time that this system could have been built.

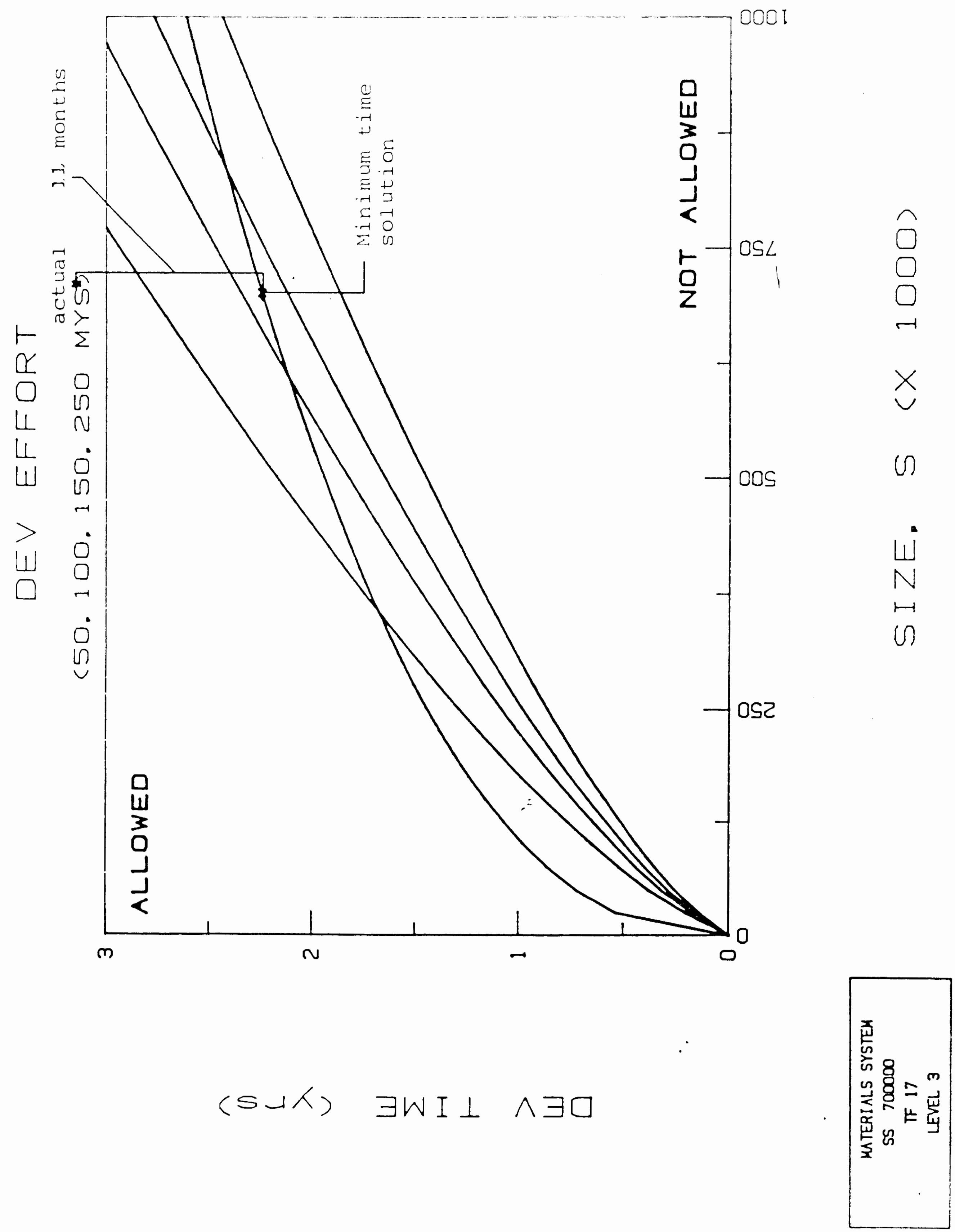
To build the system in the minimum time would require 82 people. This is not the organizational style -- a few good people is their approach.

SENSITIVITY PROFILE FOR MINIMUM TIME SOLUTION

	SOURCE STMTS	MONTHS	MANMONTHS	UNINFLATED COST (X 1000)
-3 STD DEV	600000.	25.2	1178.	4907.
-1 STD DEV	666667.	26.4	1349.	5619.
MOST LIKELY	700000.	27.1	1434.	5927.
+1 STD DEV	733333.	27.5	1524.	6352.
+3 STD DEV	800000.	28.5	1705.	7103.

A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

TOTAL MANMONTHS	1434	IN NORMAL RANGE
PROJECT DURATION (MONTHS)	27.1	IN NORMAL RANGE
AVERAGE # PEOPLE	53	IN NORMAL RANGE
PRODUCTIVITY (LINES/MM)	488	IN NORMAL RANGE


DEV EFFORT
(50. 100. 150. 250 MYS)
11 months
actual
Minimum time
solution
ALLOWED
NOT ALLOWED
SIZE, S (X 1000)
DEV TIME (yrs)
0
250
500
750
1000
0
1
2
3
MATERIALS SYSTEM
SS 700000
TF 17
LEVEL 3

!!!

MANLOADING

!!!

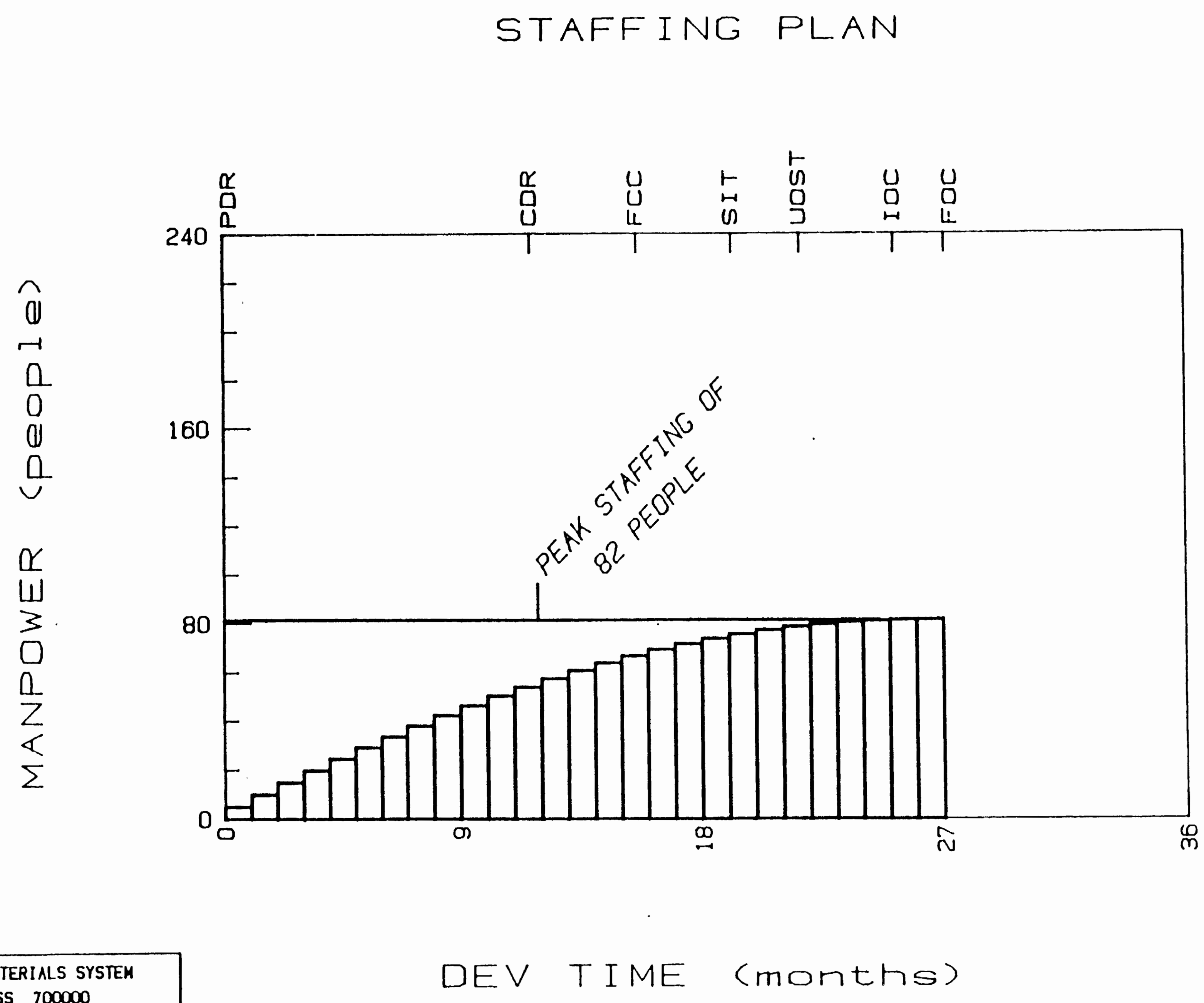
MATERIALS SYSTEM

THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT (AND STANDARD DEVIATION) REQUIRED FOR DEVELOPMENT. THESE VALUES ARE BASED ON A DEVELOPMENT TIME OF 27.1 MONTHS AND A TOTAL DEVELOPMENT EFFORT OF 1434.2 MANMONTHS.

STAFFING PLAN

MONTH	PEOPLE/MONTH	STD DEV	CUMULATIVE MANMONTHS	CUM STD DEV
JAN 79	2	.4	2	
FEB 79	7	1.3	10	1
MAR 79	12	2.1	22	2
APR 79	17	2.9	39	4
MAY 79	22	3.7	62	6
JUN 79	27	4.4	88	9
JUL 79	31	5.0	120	12
AUG 79	36	5.6	155	16
SEP 79	40	6.2	196	20
OCT 79	44	6.7	240	24
NOV 79	48	7.2	288	29
DEC 79	52	7.6	340	34
JAN 80	56	8.0	396	40
FEB 80	59	8.4	455	45
MAR 80	62	8.7	518	52
APR 80	65	9.0	583	58
MAY 80	68	9.2	651	65
JUN 80	71	9.4	722	72
JUL 80	73	9.5	794	79
AUG 80	75	9.7	869	87
SEP 80	76	9.8	945	94
OCT 80	78	9.8	1023	102
NOV 80	79	9.8	1102	110
DEC 80	80	9.9	1183	118
JAN 81	81	9.8	1263	126
FEB 81	81	9.8	1345	134
MAR 81	82	9.7	1426	142
APR 81	41	4.8	1467	150

This is more people than they usually use on a project.



MATERIALS SYSTEM
SS 700000
TF 17
LEVEL 3

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                              DESIGN TO COST
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                             MATERIALS SYSTEM


THE BEST ESTIMATES OF THE MINIMUM TIME AND CORRESPONDING EFFORT AND COST TO
DEVELOP YOUR SYSTEM ARE:

             MINIMUM TIME:          26.99 MONTHS
             EFFORT:                 1430 MANMONTHS
             COST:                   5967 (X 1000 $)

ENTER DESIRED DEVELOPMENT EFFORT IN MANMONTHS
?
384 <---
```

They actually built the system in 38 months for 384 manmonths.

Let's try these numbers

```
                          NEW DEVELOPMENT TIME
                            MATERIALS SYSTEM

                                              MEAN    STD DEV
                                              ----    -------

         NEW DEV TIME (MONTHS)               37.45      1.24
         NEW DEV EFFORT (MANMONTHS)            384        40
         NEW DEV COST (X 1000)                1600       204
```

This is within two weeks of what actually happened.

```
YOUR FILE IS NOW UPDATED WITH THESE NEW PARAMETERS.  RUN MANLOADING AND CASHFLOW
TO SEE HOW THESE SAVINGS CAN BE REALIZED.


A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

   TOTAL MANMONTHS                  384        < NORMAL RANGE
   PROJECT DURATION (MONTHS)       37.5       IN NORMAL RANGE
   AVERAGE # PEOPLE                  10        < NORMAL RANGE
   PRODUCTIVITY (LINES/MM)         1823        > NORMAL RANGE
```

NOTICE THE FLAGS

Less than normal effort

Less than normal # people

Greater than normal productivity

THESE ARE GOOD FLAGS !!

MANLOADING

MATERIALS SYSTEM

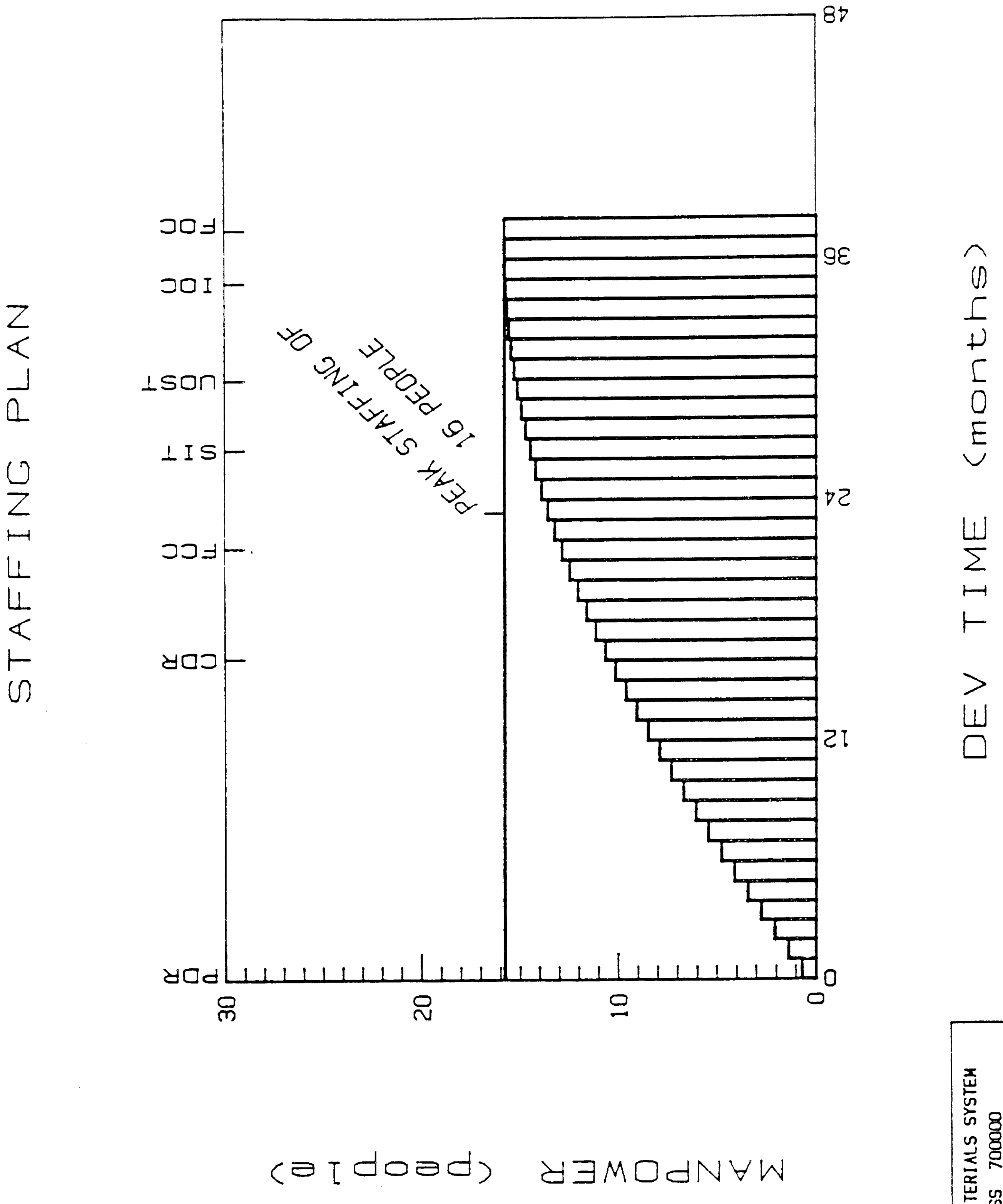
THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT (AND STANDARD DEVIATION) REQUIRED FOR DEVELOPMENT. THESE VALUES ARE BASED ON A DEVELOPMENT TIME OF 37.5 MONTHS AND A TOTAL DEVELOPMENT EFFORT OF 384.0 MANMONTHS.

STAFFING PLAN

MONTH	PEOPLE/MONTH	STD DEV	CUMULATIVE MANMONTHS	CUM STD DEV
JAN 79		.1		
FEB 79	1	.2	1	
MAR 79	2	.3	3	
APR 79	2	.5	6	1
MAY 79	3	.6	9	1
JUN 79	4	.7	12	1
JUL 79	4	.8	17	2
AUG 79	5	1.0	22	2
SEP 79	6	1.1	28	3
OCT 79	6	1.2	34	4
NOV 79	7	1.3	41	4
DEC 79	8	1.3	49	5
JAN 80	8	1.4	57	6
FEB 80	9	1.5	66	7
MAR 80	9	1.6	75	8
APR 80	10	1.7	85	9
MAY 80	10	1.7	96	10
JUN 80	11	1.8	106	11
JUL 80	11	1.8	118	12
AUG 80	12	1.9	130	14
SEP 80	12	1.9	142	15
OCT 80	13	2.0	155	16
NOV 80	13	2.0	168	17
DEC 80	13	2.0	181	19
JAN 81	14	2.1	195	20
FEB 81	14	2.1	209	22
MAR 81	14	2.1	223	23
APR 81	15	2.1	238	25
MAY 81	15	2.2	253	26
JUN 81	15	2.2	268	28
JUL 81	15	2.2	283	29
AUG 81	15	2.2	298	31
SEP 81	16	2.2	314	33
OCT 81	16	2.2	330	34
NOV 81	16	2.2	345	36
DEC 81	16	2.2	361	38
JAN 82	16	2.1	377	39
FEB 82	8	1.1	385	41

The actual staffing

This plan peaks at 16 people. (This <u>is</u> the typical organizational styl[e] for this company)



MATERIALS SYSTEM
SS 700000
TF 1.7
LEVEL 3

BIOGRAPHICAL SKETCH

LAWRENCE H. PUTNAM

Larry Putnam is President of Quantitative Software Management, Inc., a firm specializing in software cost estimating and life cycle management. Mr. Putnam has had extensive experience in industry and government in planning the quantitative aspects of software life cycle management including cost, schedule and manpower determination for development, and for control of the process during operations and maintenance. Mr. Putnam recently worked for General Electric Company as manager of system technologies. Prior to that, he was special assistant to the Commander, US Army Computer Systems Command, special assistant to the Assistant Secretary of the Army, Financial Management, and special assistant to the Director Army Automation, in which positions he developed and implemented systems to plan, budget and control large scale software systems. Mr. Putnam is a member of IEEE, IEEE Computer Society, AIAA and the Society of the Sigma Xi. He is also a member of the Society of Management Information Systems, a member of the International Society of Parametric Analysts (ISPA), and a member of the Board of Directors of the Baltimore and Washington Chapter of ISPA. He holds a master's degree in physics and the Naval Postgraduate School and a bachelor of science degree from the United States Military Academy.