JOSS: EXPERIENCE WITH AN EXPERIMENTAL COMPUTING SERVICE FOR USERS AT REMOTE TYPEWRITER CONSOLES

J. C. Shaw

May 1965

P-3149

JOSS: EXPERIENCE WITH AN EXPERIMENTAL COMPUTING SERVICE FOR USERS AT REMOTE TYPEWRITER CONSOLES

J. C. Shaw*

The RAND Corporation, Santa Monica, California

INTRODUCTION

In discussions of on-line versus off-line use of computers, attention frequently focuses on relative costs to the neglect of relative benefits. An experimental computing service, at remote typewriter consoles, was made available to the staff of The RAND Corporation to explore the value of on-line use of a computer. JOSS† (Johnniac Open-Shop System) monitors ten typewriters and serves up to eight users concurrently. It has been in daily use since January 1964. (An earlier version saw limited use beginning in May 1963.)

---

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

This paper was presented at the IBM Scientific Computing Symposium: Man-Machine Communication, held at the Thomas J. Watson Research Center, Yorktown Heights, New York, May 3-5, 1965.

†JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

Since 1963, and even earlier, other on-line, time-shared, general-purpose computing systems have come to fruition. The largest of these, at the Massachusetts Institute of Technology [1] and at the System Development Corporation [2], provide computing services for a broad spectrum of computing requirements.

In contrast, JOSS is limited by the capacities of the Johnniac, a Princeton-class computer constructed in 1950-53. T. O. Ellis and M. R. Davis painstakingly designed a communication system to connect familiar typewriters; and the author worked hard to design a smooth language for specifying small numerical computations. We believe the resulting improved access--not raw computing power--leads users to prefer JOSS to alternative conventional computing services for many problems.

Descriptions of the Johnniac, the typewriter communication system, and the language appear elsewhere [3-7]. Here, we deal primarily with the philosophy of the system and our users' experiences with it.

HARDWARE

The existence of an in-house computer nearing the end of its useful life provided the opportunity to experiment. Indeed, the Johnniac was used experimentally as

soon as commercial computers became better able to meet RAND's computing needs. The 4096 word magnetic-core memory, 12,288 word magnetic drum, and an austere order code presented a stiff challenge to produce any workable on-line time-shared system. It was necessary to construct a communication system to monitor and control remote typewriters and to buffer data transmissions. (In modern equipment, of course, this communication requirement may, in some cases, be better met by interrupting the central processor than by using a communications processor.)

The typewriter chosen was the IBM Model 868 with a paging control. (Teletypewriters were judged unsatisfactory and the IBM Selectric appeared too late for consideration.) Cooperative maintenance has overcome the initial troubles with the typewriters. The character set was carefully chosen to preserve all the standard punctuation marks. However, brackets and the symbols for arithmetic operations and numerical relations replaced the less frequently used special characters. A two-color ribbon switches automatically to black for output and to green for input. These features facilitate typing and reading the JOSS language and contribute to the philosophy that the user shall interact with JOSS as with a computing aide, using a natural, high-level language at all times.

SOFTWARE

To carry out the philosophy of presenting JOSS to the user as a computing aide and the only active agent with which he communicates, it was necessary to "hide" the Johnniac from the user and to present instead, the image of a person interpreting instructions and remaining in control of the situation, no matter how senseless those instructions may be. This contrasts with a code-checking session in which one is concerned with machine language instructions and representations of objects at the bit level. There one is reminded of nothing so much as an experimenter carefully setting up an elaborate electronic apparatus, then throwing the switch for a smoke test--because the machine language interpreter is prepared to manipulate only pieces of complex representations. The programmer who tries to repair the damage of a first test in preparation for a second is really doing it himself, because the available interpreter operates at too low a level for him to get any feeling of linguistically directing an agent. This effect is so strong that in programmers' jargon routines are personified as active agents, seldom the machine language interpreter.

Not so with JOSS. The programs the user gives JOSS are strictly static, passive specifications of computing and typing processes that JOSS carries out (Fig. 1). If the user specifies a process imperfectly, JOSS will call it to his attention with an error message and stand by to help with the changes. Frequently the user can direct JOSS to make the correction and continue without having to start over at the beginning of the program. Sometimes

```
*   PRIME NUMBERS

1.1 Do part 2 for d = 2(1)ip[sqrt(x)] if x > 3.
1.2 Type x if x ≠ 0.

2.1 Set x = 0 if fp(x/d) = 0.

Do part 1 for x = 2(1)50.
        x =      2
        x =      3
        x =      5
        x =      7
        x =     11
        x =     13
        x =     17
        x =     19
        x =     23
        x =     29
        x =     31
        x =     37
        x =     41
        x =     43
        x =     47

Type all steps.

1.1 Do part 2 for d = 2(1)ip[sqrt(x)] if x > 3.
1.2 Type x if x ≠ 0.

2.1 Set x = 0 if fp(x/d) = 0.

Delete all.
```

Figure 1

the user directs JOSS to carry out a sensible process but not the one intended. Then he himself must detect this from JOSS's actions and results.

JOSS lacks the problem-solving capacity to carry on a sophisticated conversation. For its comments, it simply selects from a stock of forty "canned" messages. But the timeliness and appropriateness of its remarks give a feeling of interacting with a person. After a year's experience with neutral, impersonal messages, we introduced personal pronouns and some informal diction (Fig. 2). This was done not facetiously, but rather to reinforce the model. No objections to the more personal messages have been voiced.

JOSS not only permits, it requires, high-level interaction between man and machine. The user has no way of referring to the internal representation of any of the objects JOSS deals with: value assignments, steps, or forms. The details of the representation are completely sealed off. Thus, except for hardware malfunctions, no JOSS "experts" are required to explain the behavior of the system in terms of machine-level operations. Inside knowledge is of no advantage to the user.

```
1.1 Set x(1) = [-b+sqrt(b*2-4•a•c)] / [2•a].
1.2 Set x(2) = [-b-sqrt(b*2-4•a•c)] / [2•a].
1.3 Type x(1), x(2) in form 1.
Set b = -1.
Set c = -6.
Do part1.1.
Eh?
Do part 1.1.
Error above:  Part number must be integer and 0<part<10*9.
Do part 1.
Error at step 1.1:  a is undefined.
Set a = 0.
Go.
Error at step 1.1:  I have a zero divisor.
Set a = -1.
Go.
Error at step 1.1:  I have a negative argument for sqrt.
Set a = 1.
Go.
Error at step 1.3:  I can't find the required form.
Form 1:
a =___   b =___   c =___   roots:    .____    .____
Go.
Error at step 1.3:  I have too few values for the form.
1.3 Type a, b, c, x(1), x(2) in form 1.
Go.
Error at step 1.3:  I can't express value in your form.
Form 1:
a =___   b =___   c =___   roots: ___._______.____
Go.
Error at step 1.3:  Your fields run together in the form.
Form 1:
a =___   b =___   c =___   roots: ___.____ ___.____
Go.
a =  1   b = -1   c = -6   roots:   3.0000  -2.0000
Do part 1 for c = -1(1)-4.
Error above:  Illegal set of values for iteration.
Do part 1 for c = -1(-1)-4.
a =  1   b = -1   c = -1   roots:   1.6180   -.6180
a =  1   b = -1   c = -2   roots:   2.0000  -1.0000
a =  1   b = -1   c = -3   roots:   2.3028  -1.3028
a =  1   b = -1   c = -4   roots:   2.5616  -1.5616
2.1 Do step 2.1 for i = 1(1)100.
Do step 2.1.
I need more storage space.
Delete step 2.1.
Go.
It's a mess.  Let's start over.
Go.
Error above:  You haven't told me to do anything yet.
Do part 1 for c = -5.
a =  1   b = -1   c = -5   roots:   2.7913  -1.7913
Please wind up your work and turn off as soon as possible.
Delete all.
```

Figure 2

JOSS hides the binary nature of the Johnniac by performing arithmetic such that the user sees floating decimal operations with nine digits. No conversion errors are introduced on input or output, and internal arithmetic is familiar. The user may ask JOSS to add .1 ten times and the result is indeed 1. Exact arithmetic is performed so long as results do not exceed nine decimal digits. Longer results are rounded in the conventional way.

The user directs JOSS in three basic activities: numerical processing, editing, and typing. The language contains elements to facilitate all three. The specification of forms for typing out results is done by presenting JOSS with a pattern for the output line. This is only one component of the many that are used in sophisticated report generators. The remarkable thing about these forms is the general adequacy of just two types of field specification: one for fixed-point output and one for scientific notation. If the user fails to specify a form for the output, JOSS chooses a reasonable one: JOSS types in fixed-point form if the magnitude of the result is between .001 and 1,000,000; otherwise, JOSS types in scientific notation.

TRAINING

The use of the JOSS system is best taught by example. Though several documents and a motion picture show the uses of JOSS in standard types of numerical work (Fig. 3 serves as a check list of what the system can do), an hour's demonstration at a typewriter console is still the most effective introduction. With such a demonstration and access to the documents, a new user enters a phase in which he can use the system productively for solving simple problems. Meanwhile he familiarizes himself with the full power of the system by informal exploring.

The user receives no formal description of the language, since such a description would be at least as difficult to learn as the JOSS language itself. A JOSS representative (not a computer expert) is available in each department to assist new users.

Most of the nearly 200 JOSS users had no prior direct programming experience. Those who did betrayed this fact by the style in which they first used JOSS. They frequently worked on a program till they felt that all eventualities were covered before telling JOSS to do any processing.

JOSS ONE-PAGE SUMMARY 1/12/65

DIRECT or INDIRECT

```
Set x=a.

Do step 1.1.
Do step 1.1 for x = a, b, c(d)e.
Do part 1.
Do part 1 for x = a(b)c(d)e, f, g.

Type a,b,c,_.
Type a,b in form 2.
Type "ABCDE".
Type step 1.1.
Type part 1.
Type form 2.
Type all steps.
Type all parts.
Type all forms.
Type all values.
Type all.
Type size.
Type time.
Type users.

Delete x,y.
Delete all values.

Line.

Page.
```

INDIRECT (only):

```
1.1 To step 3.1.
1.1 To part 3.

1.1 Done.

1.1 Stop.

1.1 Demand x.
```

DIRECT (only):

```
Cancel.

Delete step 1.1.
Delete part 1.
Delete form 2.
Delete all steps.
Delete all parts.
Delete all forms.
Delete all.

Go.

Form 2:
dist. = .....  accel. = __.___

x=a
```

RELATIONS:

= ≠ ≤ ≥ < >

OPERATIONS:

+ - · / * () [] ||

CONDITIONS:

if a<b<c and d=e or f≠g

FUNCTIONS:

```
sqrt(a)   (square root)
log(a)    (natural logarithm)
exp(a)
sin(a)
cos(a)
arg(a,b)  (argument of point [a,b])
ip(a)     (integer part)
fp(a)     (fraction part)
dp(a)     (digit part)
xp(a)     (exponent part)
sgn(a)    (sign)
max(a,b)
min(a,b,c)
```

PUNCTUATION and SPECIAL CHARACTERS:

. , ; : ' " _ # $?
___.___ indicates a field for a number in a form.
..... indicates scientific notation in a form.
# is the strike-out symbol.
$ carries the value of the current line number.
* at the beginning or end kills an instruction line.
Brackets may be used above in place of parentheses.
Indexed letters (e.g. v(a), w[a,b]) may be used above in place of x, y.
Arbitrary expressions (e.g. 3·[sin(2·p+3)-q]+r) may be used above in place of a, b, c, ...

Figure 3

The novice has no conflicts with other programming systems, and so has an easier time. Even grade-school children have shown an ability and eagerness to use JOSS on problems they understand.

The JOSS language is easy to learn. During the development of JOSS, its use was restricted to a select group of cooperative users who were asked to test the system during a period of rapid change. Near the end of that period, an unauthorized senior staff member was observed using JOSS before any training program had been set up. It turned out that he was taught by another unauthorized senior staff member who was taught by still another unauthorized senior staff member who learned by looking over the shoulder of an authorized user.

EXPERIENCE

JOSS users face three serious limitations in the experimental system: frequent malfunctions in the Johnniac; an irregular schedule; and no provision for filing programs for use in a later session. Speed limits the user only occasionally.

The Johnniac performs perfectly about 95 per cent of the time scheduled for JOSS service. This is perhaps

better than one should expect of an aging vacuum-tube computer. In conventional service it is adequate, but in on-line work the system usually "crashes" every other day. The loss is limited by the size constraints on programs (about three typewritten pages), but any disruption is irritating.

JOSS is incompatible with other uses of the Johnniac. No background processing occurs to absorb otherwise idle time. At the beginning of the experiment, we expected to dedicate the computer to JOSS service. But other experimental work led to important applications that could not be "bumped." So, the regular schedule for JOSS is only from 10 a.m. to 6 p.m., Monday through Friday.

The Johnniac has no magnetic tape or disk file for program storage. We rejected paper tape punches and readers at the remote stations as expensive and inconsistent with the desired central files. We also rejected a punched card file because it would require an operator, and punching and reading would interfere with the basic service.

These shortcomings in reliability, schedule, and filing should be kept in mind while interpreting the following statistics for a recent week that was free of malfunctions.

Statistics for April 1-7, 1965, 10:00 a.m. to 6:00 p.m. (40 hr)

- o JOSS worked for 77 different users in the week.
- o On the average day, 31 users conducted 49 sessions.
- o The sessions averaged 58 minutes, with 50 per cent less than 35 minutes.
- o The system was saturated with 8 users 18 per cent of the time, and 4 per cent of the time someone was waiting in the queue.
- o There were 5.7 simultaneous users, on the average.
- o The input rate was 1.9 lines per minute per user. The output rate was 3.9 lines per minute per user. That is, a line of input or output every ten seconds for every user.
- o JOSS worked on user blocks in core memory 57 per cent of the time. (This tends to 75 per cent during busy periods.)

The statistics are objective though indirect evidence of the usefulness of JOSS. Direct but subjective evidence is found in the comments of the users. One said he was "ready to give up the slide rule." Another mathematician claimed to have used the service for "everything from finding the product of two real numbers to solving fifteen

simultaneous transcendental equations" but then told how, on one problem, the easy access to computing power led him to postpone a more thoughtful approach that finally succeeded. A senior electronics engineer, arguing for a replacement of the Johnniac, wrote:

> JOSS is becoming essential to our output like paper, or coffee. It speeds up calculations; makes it easy to try experiments. It is the greatest, when it works. I have heard it compared favorably to beer and referred to as RAND's most important fringe benefit. People adjust their lives to fit around JOSS...No use coming to RAND before 10:00 am when JOSS arrives, in fact noon or after 5:00 pm is a better time, JOSS is less busy. When JOSS starts typing answers, the titillating pleasure is equalled only by the ensuing anguish when JOSS breaks off into jibberish or goes away commending your program to oblivion. We can hardly live with JOSS, but we can't live without it. We're hooked.

CONCLUSION

Sixteen months should be long enough for the novelty of JOSS service to have worn off. Most RAND staff members having at least occasional computing problems have been exposed to the system. The statistics developed from the minute-by-minute log of activity show that many users find the investment of their time and effort in the on-line direction of JOSS to be rewarding.

At this point, several challenges arise:

- To take the system out of the experimental stage and into a regular economical service on modern reliable equipment;
- To extend input-output capabilities to devices other than typewriters;
- To extend the range of application beyond small numerical problems while preserving the best features of JOSS.

ACKNOWLEDGMENTS

The JOSS experiment involved many individuals and could not have attained its modest goals without such contributions as: the stimulus from A. Newell; the authorization by P. Armer and W. H. Ware; the communications development under T. O. Ellis and M. R. Davis; the project management of K. W. Uncapher; the maintenance efforts of R. I. Yoshimura and crew; the programming assistance of Leola Cutler and Mary Lind; the instruction of users by C. L. Baker, G. E. Bryan, and F. J. Gruenberger; and, certainly not least, the enthusiasm of the first users--O. A. Gross, N. Z. Shapiro, W. L. Sibley, A. C. Smith, and J. D. Williams.

REFERENCES

1. Corbato, F. J., et al., The Compatible Time-Sharing System: A Programmer's Guide, The Massachusetts Institute of Technology Press, Cambridge, Massachusetts, 1963.

2. Coffman, E. G., Jr., J. I. Schwartz, and C. Weissman, "A General-Purpose Time-Sharing System," AFIPS Conference Proceedings (1964 SJCC), v. 25, Spartan Books, Baltimore, Maryland, 1964, pp. 397-411.

3. Baker, C. L., JOSS: Scenario of a Filmed Report, The RAND Corporation, RM-4162-PR, June 1964.

4. "The JOSS System: Time-Sharing at RAND," Datamation, Vol. 10, No. 11, November 1964, pp. 32-36. (This article is based on Ref. 3.)

5. Shaw, J. C., "JOSS: A Designer's View of an Experimental On-Line Computing System," AFIPS Conference Proceedings (1964 FJCC), v. 26, Spartan Books, Baltimore, Maryland, 1964, pp. 455-464; also, The RAND Corporation, P-2922, August 1964.

6. -----, JOSS: Examples of the Use of an Experimental On-Line Computing Service, The RAND Corporation, P-3131, April 1965.

7. Shaw, J. C., JOSS: Conversations with the Johnniac Open-Shop System, The RAND Corporation, P-3146, May 1965.