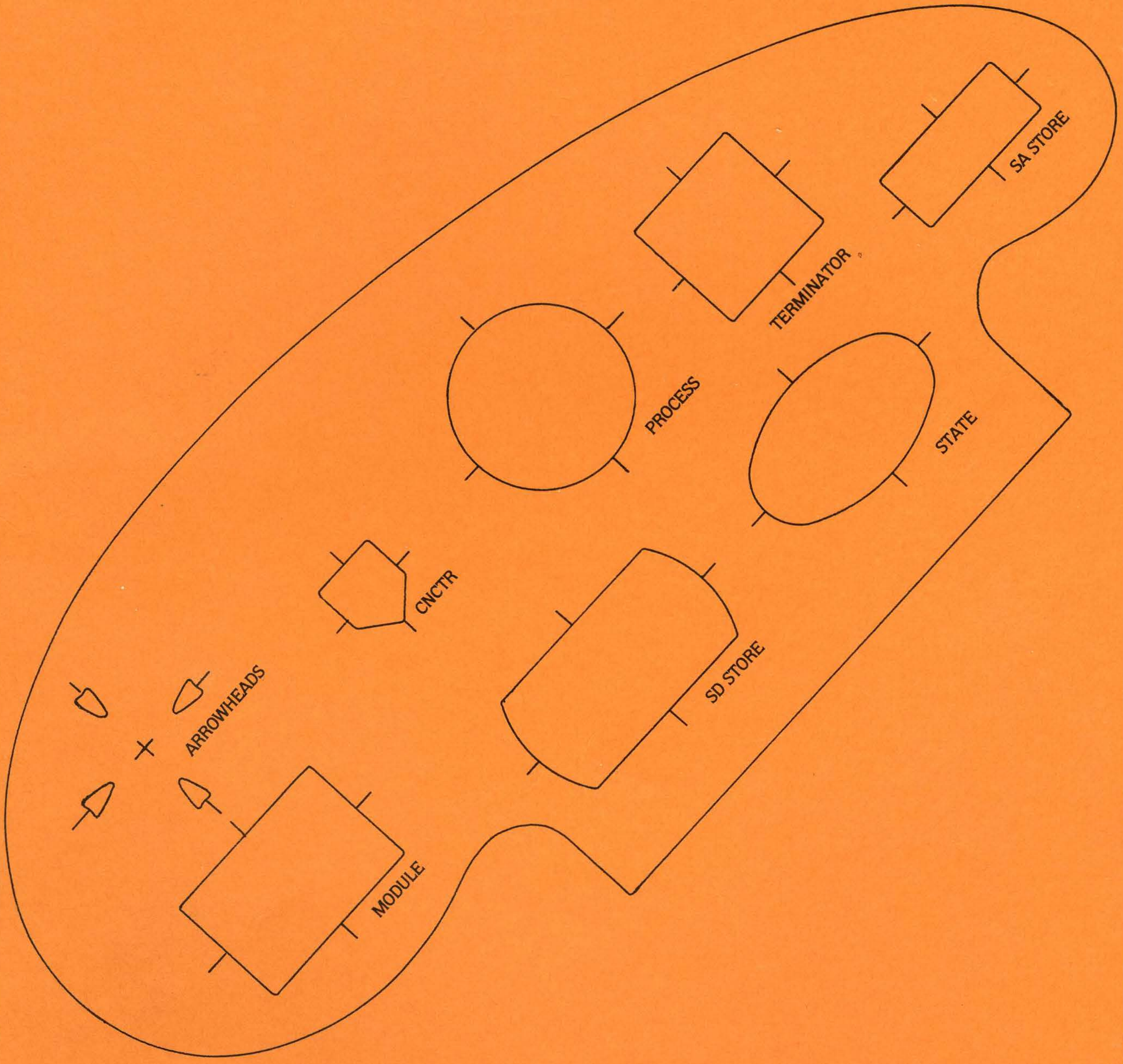
"THE HEAP"

October 1984 Issue



LANGUAGES AND TOOLS SIG

Printed in the U.S.A.

"TABLE OF CONTENTS"

TITLE	PAGE
Notes From the Editor: Introductions, Excuses, and Pleas for Contributions. - Al Folsom, Fischer & Porter Co.	2
Notes From the SIG Chair - Kathy Hornbach, Lear Siegler	4
Report on the International Software Engineering Conference - Kathy Hornbach, Lear Siegler	6
MATRIX, A Language Using Decision Tables - Joe Brugman, Plessey Peripheral Systems	10
LTSIG Wishlist - Al Folsom, Fischer & Porter Co.	18
DEC/TEST Manager - Lee Rodabaugh, Digital Equipment Corp.	27
Converting to PDP-11 Pascal	30
LTSIG Sessions for Anaheim	32
ADA[*] Certification and Validation: First Steps to a Production ADA System. - Ronald F. Brender, Digital Equipment Corporation	35
The Languages & Tools "Mushroom" The only complete SA/SD Tool, available from the LTSIG. Kathy Hornbach, Lear Siegler	40
The LTSIG and Standards Activities - Jay Wiley, Bechtel Power Corporation	41

---

* ADA is a trademark of the DoD.

My name is Al Folsom, and, if you're reading this, my first attempt at newsletter editing has not been an abject failure. I took on this role at the Cincinatti Symposium, when John Barr was forced to give it up due to work constraints. My primary goals will be to get the newsletter published on a regular and timely basis, and to insure that the articles published are of maximum interest to the SIG membership.

I hope I've made a good start with the current issue of "THE HEAP". Our Standards Coordinator, Jay Wiley, has contributed a synopsis of software engineering standards activities. If you are interested in this area, Jay has provided a cornucopia of useful information. In addition, we have articles on a decision table based programming language, a description of the new DEC/TEST Manager, a report on the International Software Engineering Conference, and a variety of information relating to the Anaheim Symposium.

Speaking of the Anaheim symposium, it is my hope that this reaches you sometime <u>before</u> the start of the symposium. As I type this, it is late September, and I hope to ship the newsletter to the DECUS office within the next two days. Part of my learning curve, however, will be to get a feel for the lead times involved in producing the newsletter. The Languages and Tools SIG is sponsoring a wide variety of sessions at this symposium, including an emphasis on the ADA language. I have included a list of sessions, but in addition to that list, of particular interest and note is that Jean Ichbiah, a principal architect of the ADA programming language, has agreed to give the Languages and Tools keynote address, certainly a session that no one interested software engineering should miss. Since ADA will be a central topic of the symposium, I have included slides from a presentation by Ronald F. Brender of Digital, concerning the validation process for ADA. While these pages may be a little difficult to read due to the duplicating process, they are certainly worth looking over if you are interested in the development of the language.

The SIG itself has undergone some changes in the last few months. For one thing, we are no longer the "Structured Languages SIG", but are now titled the "Languages And Tools SIG", abbreviated LTSIG. This is to reflect a concern not only with Languages, but with the whole realm of tools and products which affect the software engineering environment. Digital has been filling a void in this area with tools such as CMS, MMS, DEC/TEST manager, and so forth, and we wish to reflect the growing importance of these types of productivity tools. In addition to the name change, we have had some personnel changes in the steering committee. Primarily, our new SIG chair is Kathy Hornbach of Lear Siegler. If you attended the Cincinnati symposium, or have been reading your DECUScopes, you may recognize Kathy as the refereed papers winner. Kathy's primary interest is in the area of productivity tools.

Here is a partial list of steering committee members, feel free to contact any of us if you have questions or suggestions, and if you do attend the Anaheim symposium, try to look us up in the LTSIG suite.

Kathy Hornbach
LTSIG Chair
Lear Siegler/Instrument Div
4141 Eastern SE MS 121
Grand Rapids, MI 49508

Barb Chase
Vice Chair
Hughes Aircraft Company
P.O. Box 92426
Bldg R1, MS D340
Los Angelos, CA 90009

Alan L. Folsom, Jr
Newsletter Editor
Dept 431, Corporate Cntr
Fischer & Porter Co.
200 Witmer Rd.
Horsham, PA 19044

Mark Katz
Session Notes Editor
GTE Sylvania
77 A Street
Needham, MA 02194

Jim Livingston
Product Planning/Past Chair
Measurex Corporation
1 Results Way
Cupertino, CA 95014

J.R. Westmoreland
Symposium Coordinator
6748 Acoma Road
Midvale, UT 84047

As you will see in the article dealing with the LTSIG's Wishlist, one of our members' most common requests is for an introductory issue of the newsletter, dealing with some of the public domain software we have distributed in the past, and possibly surveys or lists of other available software. I hope to do this in the next issue, and would like to take this time to solicit articles in that vein. In addition, other articles of general interest would be greatly appreciated. Submissions preferably should be in machine readable format, although we can use hardcopy. (all the articles for this issue were submitted in hardcopy format). If you would like to submit material, write me, or call at (215) 674-7154, and we can arrange for a suitable media or format. Also, if you have suggestions for types of articles which would make the newsletter more valuable to you, I would like to hear from you. It has been suggested that we should print summaries of sessions from the symposia, for those people who could not attend. Would this be of interest to you, or would you prefer some other type of article? Please let me know.

Finally, I would like to change the name of the newsletter. "THE HEAP" has unsavory connotations, and is a holdover from the days when this was the Pascal SIG. If you have suggestions for a new name, send them to me, and they will be judged by a completely partial random jury of assorted weird programmer types. I'll send an "LTSIG Mushroom" to whoever submits the winning name.

Looking Towards Anaheim

Now is the time to start thinking about attending the Fall DECUS Symposium, at the Anaheim Convention Center, December 10 through 14. There are a record number of sessions being sponsored by the Languages and Tools SIG this Symposium -- you will find a complete list elsewhere in this issue.

Sessions

Coming out with version 4 of VMS are new versions of many of the DEC languages and tools -- such as Fortran, CMS, MMS, DEC/Test Manager, editors, and VAX Debugger -- and sessions are scheduled to bring you up to date on all of these. Other sessions will highlight C, Pascal, and PL/1. Tutorials will introduce attendees to effective use of tools for improving software development. User papers on experiences with Digital and third party languages/tools are also offered. For the first time, there will be a Languages and Tools Question and Answer session, with a panel of Digital developers and expert users; ready and hopefully able to answer your most challenging technical questions.

Spotlight on Ada

The SIG is highlighting the Ada[*] programming language this symposium -- a language that will have a tremendous impact on software development in the coming years. We are proud to announce the Languages and Tools Keynote Address will be given by Jean Ichbiah, one of the principal architects of the Ada language. In addition, Digital will be giving several technical sessions on their VAX Ada compiler; and we have many presentations by other companies working on Ada compilers or programming in Ada.

Seminars

The SIG is sponsoring three Pre Symposium Seminars. They deal with three different facets of software development, and all promise to be excellent.

- o Portable Software/Rapid Prototying with Software Tools VOS -- The Software Tools Virtual Operating System (VOS), a public domain software package, provides an excellent environment for the development of software, particularly software that must run on different types of hardware and/or operating systems. The UNIX-like[**] package contains over 100 utilities and several hundred library routines which form a powerful toolbox for the developer engaged in rapid prototyping or production of portable software.

- o Artificial Intelligence -- This course will be an introduction to the capabilities of Artificial Intelligence, as they are today and as they may develop in the future. It will attempt to remove some of the

----

[*] Ada is a trademark of the DoD

[**] UNIX is a trademark of Bell Labs

mystique surrounding AI, by covering the standard approaches used in implementing AI solutions, including expert systems, natural language understanding, robotics, and vision. It also includes a discussion on what it takes to actually manage and develop a product-quality AI system, which will be valuable to companies thinking of acquiring or developing AI products.

o Implementing a Software Development Environment -- This seminar will discuss software development tools for all software life cycle phases, and how to integrate them into an overall software development environment. Specific types of tools, how they are used, and sources for tools on DEC equipment will all be covered in detail, as will justifying tool purchase and promoting use of tools within an organization.

Other Symposium Activities

We are planning on repeating our popular Tool Sources handout -- listing suppliers of software tools for requirements, design, coding, verfication, documentation, ... that run on DEC hardware.

In addition to the formal sessions we present, we have user-driven 'Birds of a Feather' meetings where people with like interests discuss a specific topic; and a campground and SIG suite where you can meet the SIG leadership, Digital developers, and other DECUS members with similar interests. to get the most out of the week.

Be sure to attend the Roadmap session, first thing Monday morning. There you will be filled in on special sessions of interest, introduced to Languages and Tools Steering Committee members and Digital representatives, and in general given hints on how to get the most out of the week.

In addition to the formal sessions, the Languages and Tools SIG will again be hosting a suite in the headquarters hotel, and a campground in the Convention Center. These are places where you can escape from the hectic pace for a little while, meet people interested in the same things you are, and talk with SIG leadership and Digital developers.

As you can see, many exciting things are happening in the area of Languages and Tools at Anaheim. Your attendence at sessions, and active participation in the SIG, can not only help you in your work, but will also give you a chance to shape the direction of future Digital offerings in the area of languages and tools!

-- Kathy Hornbach
Languages and Tools Chair

Report on the Internation Software Engineering Conference

This report is a brief synopsis of the Seventh International Conference on Software Engineering (ICSE), held last March in Orlando, Florida. ICSE, sponsored by the IEEE and ACM, is the leading forum for introduction of new ideas into the field of software engineering. It was attended by well over 1000 people, with a large percentage of them from Europe, Japan and elsewhere. The conference is held every 18 months, alternating on and off the North American continent. There were a lot of recognizable names there, for anyone familiar with the Computer Science field. These are the people breaking new ground, and writing papers on future directions in software engineering.

Overview

There were several marked trends that became apparent from the papers given at the conference. These are not things that will affect the ordinary software developer immediately, but they will undoubted exert a large influence over the next five to ten years (just as papers in the early seventies on Structured Analysis, source code control and UNIX[*] are of central importance now). These trends include:

- o The emphasis is on requirements - There was very little attention paid to the design and coding phases - it was almost as if these were considered solved problems. Formalized requirements definition, on the other hand, was treated as a completely open problem. In fact, the emphasis was more on defining what the problem is, as opposed to figuring out how to solve it.

- o Artificial intelligence may be the next panacea - Much as "Structured <fill-in-the-blank>" was touted as the solution to all the woes of software development a decade ago, artificial intelligence is seen by many as a cure-all. The keynote address was by Herbert Simon, a nobel prize winner and one of the pioneers in the field of AI. Many of the papers centered on AI-oriented solutions to requirements definition problems. As with the Structured Methods, there is probably much value in these techniques, but there remains much to learn, much work to get people to use them, and many new problems will inevitably appear to replace solved ones.

- o Rapid prototyping is even more popular - Rapid prototyping (the putting together of a quick "breadboard" version to try out a concept and get feedback from users) has been a hot topic for the last few years. There was much emphasis on this area again this year - executable specifications are the latest spinoff. The only real world applications of prototyping, however, seem to be in the area of user interfaces for information systems.

- o Stop the waterfall - Just as we are patting ourselves on the back for finally adopting the formalized "requirements-design-code-test"

----

[*] UNIX is a trademark of Bell Labs

software life cycle "waterfall" model, the people who proposed it in the first place change their mind. They say it really isn't the way software should be developed after all! Actually, what they really are proposing are modifications to the life cycle to reflect reality even more accurately - encompassing rapid prototyping, places for AI assistance, etc.

o <u>The generation gap widens</u> - The gap between state-of-the-art and real-world software development techniques seems to be getting bigger. The papers were on AI and executable specs and formalized proofs of correctness. The people in the audience working on real projects were struggling to get their company to recognize that there is such a thing as a software life cycle; that formalized methods are worthwhile; that code should be commented and controlled; even that interactive terminals are beneficial.

<u>SESSION DETAILS</u>

1.0 SOFTWARE DEVELOPMENT PARADIGMS

Fred Brooks (of Mythical Man Month fame), Harlan Mills (from IBM and originator of much of Structured Programming), and Tom Cheatham had a lively debate on the correct "model" for software development. Brooks advocated the traditional software life cycle, modified by rapid prototyping.

Harlan Mills put forward some surprising new concepts, based on research he is doing back at IBM. The first concept is of "Structured Data" - get rid of pointers and arrays; use the higher level constructs of queues, stacks and sets in their place. (Of course, these would eventually be implemented with pointers and arrays, but this should be hidden just as GOTO's are hidden in DO WHILEs). Based on experiments he did, he postulates that use of these high level data types reduced verification difficulty by a factor of five. If this seems somewhat far-fetched, remember that goto-less programming did too, when he first talked about it.

The second concept he discussed is that of "clean room" software development. Under this method, the development team does the design and writes the code - but goes no further than obtaining a clean compile. A separate test team, that has been developing functional tests concurrently with design and coding, runs the test set against the code. Results are returned to the development team, who must then fix the code based only on the results from the test team, get a clean compile, and resubmit it to the test team. What has happened in actual experiments is that the development team spends much more time in code walkthroughs and in desk checking their code. The overall time to implement a correctly working program has been less than on traditionally developed programs. And, to their surprise, the development team on the whole did not resent not being able to execute their own code.

Cheatham talked about the future, with AI assistance in program development, and how such a system might evolve from current capabilities. The first step is putting project knowledge in on line data bases, and coordinating that with an integrated tool set. Later stages formalize properties and attributes, and draw inferences based on data in the data base.

2.0 EXECUTABLE STRUCTURED ANALYSIS

Tom DeMarco (co-inventor of Structured Analysis) gave an interesting talk on some work he is doing in Modula-2 on the Lilith computer. He has extended the concept of pipes from UNIX. UNIX supports three standard I/O channels per process - input, output and error. DeMarco extends this concept to many ports per process by giving each port a name. Processes are then connected together by connecting named ports with a graphical editor. Each bubble is represented by an executable process. A bubble can also represent an entire lower level sequence of bubbles and data flows, just as in SA. Each data flow corresponds to a named port. This is indeed very similar to a Data Flow Diagram.

They have a primitive version of this tool working on Lilith. This is the first I have seen of an actual implementation of "executable specifications". However, as questions from the audience pointed out, it is not clear how many problems - even simple ones - would be amenable to this type of solution. DeMarco admitted it was difficult to even come up with examples of problems that could be solved with this method. There was some talk that this approach had been tried before, and dropped when it was discovered it was not all that useful.

Nonetheless, it is an interesting concept. There is a paper on it in the Proceedings.

3.0 PRODUCTIVITY FACTORS

A study by the ITT Advanced Technology Center, of several real world projects attempted to pin down which factors contributed to effective completion of a project, and to high productivity. There were wide spreads in the ranges of data for some of the factors, which was initially puzzling. For example, some projects that employed modern programming practices had comparatively high levels of programming productivity; but some projects using the same methods had relatively low productivity. However, all projects that did NOT employ modern programming practices had relatively low productivity.

The conclusion drawn is that it is not enough just to use SOME productivity techniques. To really succeed, a software project must do EVERYTHING right. Failure in one of the areas can drag all the others down with it.

4.0 CASE STUDY OF ADA INTRODUCTION

A division of GE did a pilot project on teaching Ada [**] to embedded system programmers, to better understand the problems and considerations to effectively learning the language. The conclusions they reached reinforce some

[**] Ada is a trademark of the DoD

suspicions we have had all along about ADA training. They include:

1. Most training programs stress teaching the syntax of the language. The hard part is understanding the CONCEPTS behind various parts of the language. An effective training course would include much background information on concepts like data abstraction and information hiding. They found that much training is needed, in some cases, to bring people up to speed on the concepts alone.

2. The examples given in much of the training are from computer-science-type applications. Students expressed much interest in having problems drawn from real world applications, such as sensor monitoring, navigation, etc.

3. It is important to have a compiler to try out all the things that are being learned.

4. Although most of the features of Ada were used by programmers on their first attempt, they were used in ways other than for what the language designers intended (again, better grounding in concepts would help here).

5. Many support tools are clearly needed. A language sensative editor is absolutely required - it would have save enormous amounts of time.

6. The requirements for the project were too detailed - more like than unlike a Fortran system design. Implementors found they had to abstract UP!

Summary

ICSE was a fascinating conference in many ways. In many ways it is the exact opposite of DECUS Symposia sessions, which tend to be problem-specific, immediately useful, and grounded in the real world. ICSE was theoretical and non-specific, and I had no new knowledge I could take back home and put to immediate use. BUT, on the other hand, it provided an excellent preview of the directions software engineering is likely to be taking over the next five, ten and fifteen years. While the things I learned may not be immediately useful, many of the concepts I heard talked about for the first time, will always be in the background of my thinking now, helping shape how our whole software development environment will be evolving over the next few years.

If any of you have a chance, I would recommend attending this conference, keeping in mind its goals and targeted audience. The next one is in London, England, in August, 1985. You can order the Conference proceedings for this and all previous ICSE from the IEEE.

-- Kathy Hornbach

MATRIX, a Language Using Decision Tables

by Joe Brugman

ABSTRACT

Using decision tables to direct the processing of business computer programs provides a rapid and easy method of creating computer programs. This paper describes such a language.

New languages seem to make their appearance quite often. Many are special-purpose languages; others assist the programmer by selecting routines from a library to perform various desired functions. Some are oriented away from the FORTRAN and COBOL format toward an objective or results language letting the compiler (?) decide how to obtain the desired end.

I have come across a language which is completely different from any of these. It is quite popular in Europe under several different names, but MATRIX will due as a general reference. Why Matrix? Because it is a decision table based language. The matrix of a decision table is used to determine the flow of operations to obtain the desired processing of the data. This may be handling data input and storage on a file; it may be processing that data, with other information, to produce a variety of reports; or most any other activity used in business data processing. The use of pointers, bits, addresses of variables and other types of programming aids are also features of the languages.

MATRIX has some relation to COBOL in that all variables are defined in the front of the program. The files that will be used are specified and the location, size, and type of each field that will be used in each record are defined.

Having established the tools that will be used, the program next needs to know what results you intend to produce. So a list is made up of the variables that will be used in the output. These variables are associated with a single letter or number. There are three types of output variables, and what character is used is significant. Variables which are printed and have no other significance or function are assigned the letters A through L, though others from the end of the alphabet and some other printable characters may be used if necessary to the output. Those variables which cause a break requiring a subtotal are assigned letters M through R. These totalling fields cause breaks in sequential alphabetical order with M being the major break, N the next most significant, etc. Finally, since a break generally means that a total is to be taken, the totalling fields carry single-digit designators, and if necessary, letters starting with S at the end of the alphabet.

Of course, if breaks and subtotals are to have any meaning, the data must be sorted. The sorting order is specified using variables which have been defined either as field names or as the names given working variables. Numbers, other than single precision integers must be converted to strings before sorting.

Now that we have defined what we want to output in the way of variables, the printed output is defined. (Note that output to a file is handled in the same manner as input from a file except that the data is moved into the field and the record is written instead of the reverse.) Using these assigned numbers and letters, a "picture" of each line of output is made. These pictures are labeled for their desired use. A header format and the necessary totalling formats for each break may be defined as well as an output for each record. If any formats are not defined, no problem, the program continues without printing the line or break or whatever. One of the chores of handling breaks is taken care of by the program, and that is rolling over the subtotal to the next higher level and the fields zeroed.

By now it's time to get down to massaging the data. A starting decision table, (DETAB) is used to set up initial conditions if any are desired. When it is complete, the program automatically starts processing the DETAB named RECORD. If desired, one file may be defined as a "controlling" file and will be read sequentially. Depending on what is read or the results of some operation on the data, another file may be randomly accessed and further actions taken. Finally, the results may be printed and output to a file, records updated, or other actions taken.

"What is this DETAB that does all these wonderful things?" "How is it constructed and how does it work?", I hear you asking. A decision table is familiar to students of logic; it is a short and concise way of expressing what can be very complex conditions and actions taken depending on these conditions. The conditions are posed in the form of a question: is A equal to B? This condition stub is then followed, on the same line, by the condition entry: usually yes (Y), no (N), or don't care (-).

A series of conditions are followed by actions. The action stub consists of a verb and usually a variable name to be acted upon. It may also have a second operand if appropriate. The action entry for each action stub is directly related to the conditions set forth in the DETAB. If a rule (explained presently) is satisfied, then the action, or non-action for each action entry is indicated. An "X" means "do it"; a "." means "don't." Since all the Ys, Ns, and -s appear within a regular matrix, a rule is easily defined as the condition entries in any particular column.

When a rule is satisfied, i.e. the condition entries in one column are all as specified for the data being tested, then the actions in the action entries with Xs in that rule are carriedout.

Consider this DETAB:

```
C    A    =    B           Y   Y   N   N
     A    >    C           Y   N   Y   N
A    C    MV   25.00       X   -   -   -    [ MOVE 25.00 TO C
     C    MV   10.00       -   X   -   -    [ MOVE 10.00 TO C
     C    MV   0           -   -   X   -
     CALL ADDRESS          X   X   X   -
     GOTO RECORD           X   X   X   X
```

The "C" and "A" in column 1 are aids to the compiler to signal which stub is being parsed. The comments are introduced by a "[." The "CALL" verb means to process the DETAB named ADDRESS and then return to process the next action.

An extended entry stub may contain a question mark for one of its operands. Then the value to substitute for the question mark will be found in the appropriate entry. For example:

```
C    DEPT =    ?           2   17  30  ELSE
A    AMT  MV   ?           100 50  80  -
```

If DEPT is 17, then 50 will be moved to AMT. The ELSE rule handles all records which don't satisfy any other rule.

I can run the language on my PDT (MINI MINC) under RT-11. I have used it extensively under V3B, and some under V5. There are some bugs in the program, but they can be worked around.

I also have a version that runs under RSTS/E. I have not used it, however, since V5 so I don't know how it will work under any later versions. Some day I'll try it out. but since RT-11 works with the latest version, I suspect that the RSTS may also.

To show the language, I wrote two programs to read the same data and print the results. They are necessarily short and quite simple. Both are written to run under RT-11. Although the BASIC code could be improved upon, and take less space and effort to code, it cannot come near to being as short and as simple as the MATRIX version. The file used for these programs contained entries which were ignored. It was organized in Account Number order within Area as a sequential file.

To summarize: MATRIX is a fast, simplified data processing system which can be used extensively for the selection and printing of data from computer files, and for the creation and maintenance of those files. MATRIX is straightforward to learn and quick and easy to write. Incorporating a powerful Decision Table Processor, MATRIX allows for easy definition of record selection criteria and processing requirements; it is controlled by simple parameters which define input, output, and processing required by a problem. As little or as much of the automatic processing of MATRIX as the programmer desires may be incorporated in the program.

MATRIX is a compiled language which can be used not only by the data processing staff as a general purpose utility, but also by non-technical staff and management to provide information retrieval and report generation.

MATRIX, under other names, is being used in hundreds of installations in Europe and Australia. Reports show that many of them are placing increasing reliance on the language in general development work to maximize programming output; all report substantial savings in development time, often up to 80 percent compared with COBOL.

MATRIX program logic is expressed in decision tables which are easy to understand, write and amend, and document. Decision tables normally consist of "conditions" followed by "actions", but in MATRIX, it is possible to specify "initial actions" prior to "conditions", thus reducing the number of decision tables required to perform the program logic.

In brief, MATRIX is a comprehensive file maintenance and reporting system. The facilities offered by MATRIX make it suitable for a wide range of applications. They include input validation, file creation and maintenance, selective reporting with arithmetic actions taken on desired fields, subtotals on selected field breaks, and information retrieval.

MATRIX has been designed around three fundamental principles which will increase productivity:

1. simple, and largely free-format, statements are used to define requirements

2. decision tables, a simple and powerful way of expressing logic, are used to specify, easily, concisely, and accurately, the record selection and processing requirements

3. capability of interrupting MATRIX's automatic sequence of operations to specify your own processing requirements

These provide a high level of utility characterised by simplicity of syntax and ease of understanding. Bpth accuracy and speed of programming are enhanced.

If this paper has tickled the fancy of any of you fine readers, and you would like to write, I would be happy to respond. Enclose an SASE with your letter. I also think the language could be improved and rewritten in C. Certainly the bugs could be fixed. Any comments? Write me at 15434 Janine Drive, Whittier, California 90603.

About the author

The author has a Bachelor's Degree in Engineering and a Master's Degree in Business Administration. He has been engaged in data processing for 19 years having been involved in all aspects of that field. He is currently managing the technical support and program production for Plessey Pheripheral Systems.

```
PROGRAM BALACT.BAS                3-SEP-84    1342 HOURS

1 REM
5 REM BALACT.BAS  JJB  04-MAY-84  VNC
10 REM
220 L$="    'L         'LLL          $$######.##"
240 S$="    AREA 'L TOTAL            $$######.##"
260 T$="    FINAL TOTAL              $$######.##"
280 I$="          " \ REM 10 SPACES FOR INDENT
1000 REM
1001 REM              *****  INITIALIZE  *****
1002 REM
1010 P%=0
1020 T=0
1040 T1=0
1060 T2=0
1080 DIM #1,R1$(18%)=255%
1100 OPEN "DK1:CUSTFL.DAT" FOR INPUT AS FILE #1
1120 OPEN "LP0:" FOR OUTPUT AS FILE #4
1400 Y$=R1$(1%)
1420 GOSUB 5000
1440 A$=A1$
1460 M$=M1$
1480 B$=B1$
1600 GOSUB 11000
2000 REM
2001 REM              *****  LOOP THROUGH FILE  *****
2002 REM
2020 FOR I%=2% TO 18%
2040 Y$=R1$(I%)
2060 GOSUB 5000
2100 IF M$<'9000' GO TO 2200
2150 IF M$>'9799' GO TO 2200
2180 GO TO 2800
2200 IF A$<'21' GO TO 2800
2250 IF A$>'22' GO TO 2800
2300 B=VAL(B$)
2500 IF A$=A1$ THEN  GOSUB 3000 \ IF I%=18% THEN  GOSUB 4000
2600 IF A$<>A1$ THEN  GOSUB 3000 \ GOSUB 4000
2800 A$=A1$
2820 M$=M1$
2840 B$=B1$
2880 NEXT I%
2900 GO TO 4500
3000 REM
3001 REM              *****  PRINT A LINE  *****
3002 REM
3020 PRINT #4,I$;
3050 PRINT #4,USING L$,A$,M$,B/100
3060 T1=T1+B
3100 L%=L%+1%
3120 IF L%>60% THEN  GOSUB 11000
```

```
3200 RETURN
4000 REM
4001 REM                   *****  AREA BREAK  *****
4002 REM
4020 PRINT #4 \ PRINT #4,I$;
4030 PRINT #4,USING S$,A$,T1/100
4040 T2=T2+T1
4060 T1=0
4080 PRINT #4 \ PRINT #4
4200 RETURN
4500 REM
4501 REM -                 *****  FINAL TOTAL  *****
4502 REM
4520 PRINT #4 \ PRINT #4,I$;
4540 PRINT #4,USING T$,T2/100
4600 GO TO 31000
5000 REM
5001 REM                   *****  DECODE RECORD  *****
5002 REM
5020 A1$=SEG$(Y$,3,4)
5040 M1$=SEG$(Y$,5,8)
5060 B1$=SEG$(Y$,159,168)
5100 RETURN
11000 REM
11001 REM                  *****  NEW PAGE  *****
11002 REM
11020 PRINT CHR$(12%)
11040 P%=P%+1%
11220 L%=6%
12100 PRINT #4,I$;
12120 PRINT #4,'04-MAY-84      CUSTOMER BALANCE LISTING      PAGE';P%
12140 PRINT #4
12150 PRINT #4,I$;
12160 PRINT #4,'   AREA      ACCOUNT            BALANCE'
12170 PRINT #4,I$;
12180 PRINT #4,'             NUMBER'
12200 PRINT #4
12240 RETURN
31000 REM
31001 REM                  *****  E O J  *****
31002 REM
31100 CLOSE #1
31120 CLOSE #4
```

```
*
* BALACT.MAT
*
* BALANCES BY ACCOUNT AND AREA
*
*FILE   S255    CUSTRP.DAT
*OFILE  3  A  DK1:BALACT.LST
*
*DICTIONARY
*
        LR1                         0/255
        AREA    = LR1+3/2                   [ AREA
        ACCNO   = LR1+5/4                   [ ACCOUNT NUMBER
        BALANCE = LR1+159/10                [ BALANCE
*
        DOLLARS = %%
*
*INLIST
*
        A       ACCNO
        M       AREA
        2       DOLLARS
*
*HEAD   L       CH1,2
DD-MMM-YY                CUSTOMER BALANCE LISTING                PAGE PPPP

    AREA        ACCOUNT             BALANCE
                 NUMBER
*OUT    L       1,0
     MM          AAAA            $22222.22
*OUT    M       2,1
   'AREA'MM'TOTAL'              $222222.22
*OUT    F       3,0
   'FINAL TOTAL'               $2222222.22
*DETAB RECORD
*
C       ACCNO   GE      '9000'  Y  Y  -  -  ELSE         [ IGNORE RECORDS
        ACCNO   LE      '9799'  Y  Y  -  -   -           [   9000 - 9799
        AREA    LT      '21'    -  -  Y  -   -
        AREA    GT      '22'    -  -  -  Y   -
A       DOLLARS MV      BALANCE .  .  .  .   X
        DOLLARS /       100     .  .  .  .   X
        IGNORE                  X  X  X  X   .
*
*
*STOP
*
```

```
04-MAY-84       CUSTOMER BALANCE LISTING

   AREA       ACCOUNT            BALANCE
               NUMBER

    21          1144             $1511.20
    21          1313              $151.50
    21          8643              $109.22
    21          9827              $210.80

    AREA 21 TOTAL                $1982.72


    22          3211               $16.84
    22          3270              $128.62
    22          9988               $99.88

    AREA 22 TOTAL                 $245.34


    FINAL TOTAL                  $2228.06
```

"LTSIG WISHLIST"

Those of you who have been with the SIG for some time may remember that about a year and a half ago we conducted our first Wishlist survey. For a variety of reasons, this information was not collated until early this past summer, (although we received responses as recently as last June!), which detracts somewhat from the usefulness of the responses. The results are, finally, presented here, along with the items from the original survey.

The format consisted of two lists of actions, one directed at the SIG, and one directed at DEC. Respondents were asked to cast a total of forty votes per list, with a maximum of five votes per individual item. For example, eight items could receive five votes each, twenty items could receive two votes each, or items could be prioritized as desired. Each list was independent. In addition, a brief questionnaire was included with the Wishlist.

Perhaps the most amazing result of the survey was the revelation of how many people were incapable of following the directions! Nonetheless, I sorted them out as best I could; disallowing blatently illegal responses such as casting forty votes for one item, and translating Yes/No responses to a balanced distribution of forty votes.

Many SIG's conduct surveys under a variety of names, such as Wishlists, Menus, or SIR's. The RSX SIG was, I believe, the first to do so, and has been at least moderately successful in getting responses from DEC. As far as I know, however, we are the first SIG to have a section of the Wishlist directed at the SIG itself. I hope that this will help the Steering Committee to best meet the wishes of the SIG membership.

The first four pages consist of the original items on the Wishlist. It is important to remember that these items were all submitted by members of the SIG; they are not biased in any way towards what DEC wished to hear, or what the SIG steering committee thought was appropriate.

Following that is a synopsis of the questionnaire results. These include operating systems used, types of installations, and a breakdown of languages used by the respondents and those preferred.

Finally, there are two pages representing the results of the survey, showing first the DEC directed items, and then those directed to the SIG. On each page the items are sorted first by the number of votes received, and then by numerical order. The first column shows the item number, followed by the raw total of votes received. This is then translated to a precentage of the maximum votes the item could have received, calculated by multiplying the number of respondents by five. Finally, there is a histogram representation, comparing that particular item to the one receiving the largest vote.

For this Wishlist process to be effective, it is crucial that the turn around time be reduced. to that end, we are targetting at presenting the result of the next wishlist at the Spring 1985 symposium. Items for the surveys must come from the members of the SIG, and should be sent to the Newsletter Editor. If you wish to contribute items for the next wishlist, the address is:

Alan L. Folsom, Jr.
Dept 431
Corporate Center
Fischer & Porter Co.
200 Witmer Road
Horsham, Pa.   19044

Please indicate whether the items are intended for DEC, or for the SIG. The Wishlist cannot be effective unless a reasonable number of rational items are presented to the membership. Facetious items, or suggestions which are clearly impossible (The SIG developing and maintaining and ADA compiler, for example), will not be included. Since the LTSIG is now involved with items such as CMS, MMS, and the Test Manager, items for improvements or extensions in the area of software development tools would be particularly welcome.

If at all possible, the next Wishlist will be included in the next issue of the newsletter. It is important, therefore, that items for inclusion be received as soon as possible.

COLUMN 'A'	COLUMN 'B'
--------	--------
--------	--------
--------	--------
--------	--------
--------	--------
--------	--------

I'll take one of these,
and two of those, and two of ...

DEC DIRECTED MENU ITEMS

0.1 DEC should provide a consistant, standardized structured language, and support it across all operating systems and CPU's.

0.2 Structured languages should be provided for 10/20 machines.

0.3 Structured languages should be provided for the PDP 11 computers.

0.4 A wider range of languages should be supported for RT-11.

0.5 DEC should provide Cross Compilers for the various new 16 bit micros, to facilitate program development.

0.6 Dec should provide more closely coupled compilers and debuggers, to facilitate program development in high level languages.

0.7 A common set of debugging tools should be developed, providing a unified interface across operating systems and languages.

0.8 DEC should announce when software such as loaders are changed, so that modifications can be made to SIG or customer supported software.

0.9 A standard "C" should be provided across all operating systems.

0.10 DEC should provide and support a "C" compiler for the DEC 10/20 machines.

0.11 DEC should provide and support the "C" language for RSX systems.

0.12 The "C" language should be available for the new personal computer lines.

0.13 The "C" language should be available for programming the LSI-11 machines, perhaps as a Cross Compiler package on larger CPU's.

0.14 DEC should put the DECUS "C" compiler on the distribution kits, in the same manner as TECO.

0.15 A standard Pascal should be provided across all operating systems.

0.16 Vax-11 Pascal Global variables should be fixed to allow sharing or non-sharing across seperately compiled modules.

0.17 DEC should support Pascal for RT-11.

0.18 DEC should support Pascal for RSX.

0.19 DEC should provide a Pascal Compiler for the LSI-11, which would run with limited memory. (64k)

0.20 DEC should support Pascal under RSTS, allowing linkage to Macro or Basic+2 object modules.

0.21 PDP 11 Pascal should be provided, allowing linking to RMS 11.

0.22 Fortran 77 should be supported under RT-11.

0.23 DEC should provide customer BLISS courses.

0.24 Modula 2 should be supported under RSX

0.25 Modula 2 should be supported under VMS

0.26 Modula 2 should be supported for the new Personal Computer lines.

0.27 DEC should make an ADA package availble for the VAX.

0.28 DEC should make an ADA package availble fo the PDP 11.

SIG DIRECTED MENU ITEMS

0.1 The SIG should provide an organized method of feedback to DEC on language use and problems.

0.2 The SIG should be involved in formulating language standards.

0.3 The SIG should provide information on the use of structured languages in a time critical commercial environment.

0.4 The SIG should investigate ways of taking advantage of operating system features, while maintaining operating system/implementation independence.

0.5 The SIG should develop and maintain a source code management system.

0.6 The SIG should work on developing a set of compilers for all DEC operating systems, and a unified interface to IBM and HP systems.

0.7 The SIG should provide information and surveys of Third Party Compilers.

0.8 The SIG should develop and maintain Cross Compilers for the various new 16 bit microcomputers.

0.9 The SIG should provide "C" compiler benchmarks.

0.10 The SIG should publish information in the newsletter, and otherwise address the issue, of "C" portability.

0.11 The SIG should develop language translators, such as Fortran to "C".

0.12 The SIG should provide detailed documentation on the input format to the Code Generator phase of the Decus "C" compiler, so that users could write their own code generators for other processors.

0.13 The SIG should develop a Fortran 77 that uses Pass 2 of the DECUS "C" compiler.

0.14 The SIG should support a Structured Fortran preprocessor package.

0.15 The SIG should investigate and develop support for Fortran 77 under RT11.

0.16 The SIG should maintain a Pascal for RSTS.

0.17 The SIG should investigate improvements to the MBS Pascal Compiler and Library.

0.18 The SIG should continue, and enhance, support for Pascal running under RT11 and RSX.

0.19 An RSX Praxis should be made available on the SIG tapes.

0.20 The SIG should make an ADA package available.

0.21 The SIG should address the issue of portable programming in general, especially in the area of standard or multilanguage libraries.

0.22 The SIG should publish articles in the newsletter dealing with information and experiences concerning Praxis and/or Ada like languages.

0.23 The SIG should provide an introductory newsletter, with information about Software Tools, and what is available on the SIG tapes.

0.24 The SIG should provide a comparison of available structured languages, in terms of their capabilities and available compilers.

0.25 The SIG should develop and maintain common debugging tools for all structured languages.

0.26 The SIG should encourage development of Software Tools, to be made available through the SIG tapes.

0.27 The SIG should survey to determine in what areas software tools are most needed.

0.28 The SIG should endeavor to provide faster turnaround for SIG tapes.

0.29 The SIG should provide better documentation of updates on the SIG tapes.

0.30 The SIG should investigate other means of acquiring updates.

0.31 The SIG should develop standardized libraries for various applications, such as file I/O, graphics, and terminal handling.

0.32 The SIG should maintain the largest possible range of languages for RT11.

** OPERATING SYSTEMS **

29 respondents using TOPS10 or TOPS20
96 respondents using RSX or IAS
21 respondents using RSTS
72 respondents using RT11
94 respondents using VMS
23 respondents using UNIX
21 respondents using other operating systems

** TYPE OF INSTALLATION **

73 Scientific oriented installations
24 Manufacturing oriented installations
33 Education oriented installations
27 Business oriented installations
22 Government oriented installations
38 other installations

** LANGUAGES: USED **

1	MACRO-11	1	ASSEMBLER	3	MODULA-2
1	SIMULA	2	SIMULA67	1	BLISS10
1	PL/M	1	SAL-11	1	MAINSAIL
1	BASIC+2	1	DIBOL	2	BASIC
30	NOT SPECIFIED	3	PL/1	1	FLECS
1	SAIL	3	BLISS	2	MANY
2	COBOL	3	FORTRANIV	14	FORTRAN
80	PASCAL	32	C	20	FORTRAN77
9	RATFOR				

** LANGUAGES: PREFERRED **

2	ALGOL68	1	CEDAR	1	BASIC+
1	APLSF	2	SIMULA	1	SIMULA67
1	ALGOL-60	6	FORTRAN77	1	MAINSAIL
1	BASIC+2	31	NOT SPECIFIED	1	LOGO
1	MODULA	1	FORTH	1	SAIL
25	MODULA-2	3	ALGOL	1	APL
2	BLISS	1	FLECS	6	PL/1
8	FORTRAN	38	C	26	ADA
51	PASCAL	3	RATFOR		

** LANGUAGES: SECOND CHOICE **

1	BLISS	1	MACRO-11	1	ASSEMBLER
1	SIMULA67	1	FLECS	1	LISP
1	SUPERMAC	1	BLISS36	1	CORAL-66
3	MACRO	1	PRAXIS	3	RATFOR
1	FORTH	4	PL/1	1	MODULA
1	XPL	7	FORTRAN77	1	SMALLTALK
48	NOT SPECIFIED	2	ALGOL	11	MODULA-2
1	COBOL	24	ADA	1	BASIC
1	MODULA2	37	C	7	FORTRAN
53	PASCAL				

```
   ** SORTED DEC ITEMS **


Item     Total    Percent          Percent of Largest Vote
Number   Votes    Of Max   0                        50                       100

     1     463     42.9    **************************************************
     7     416     38.5    ********************************************
    15     398     36.9    ******************************************
     6     375     34.7    ****************************************
     9     329     30.5    ***********************************
    27     291     26.9    *******************************
     5     225     20.8    ************************
    28     219     20.3    ***********************
     3     210     19.4    **********************
    14     206     19.1    **********************
    22     179     16.6    *******************
    25     169     15.6    ******************
    12     153     14.2    ****************
     8     139     12.9    ***************
    29     121     11.2    *************
    18     119     11.0    ************
    16     114     10.6    ************
     2     102      9.4    ***********
    21      95      8.8    **********
    11      94      8.7    **********
     4      91      8.4    *********
    17      88      8.1    *********
    24      82      7.6    ********
    19      79      7.3    ********
    26      78      7.2    ********
    13      71      6.6    *******
    10      47      4.4    *****
    20      43      4.0    ****
    23      31      2.9    ***

   ** DEC ITEMS **


Item     Total    Percent          Percent of Largest Vote
Number   Votes    Of Max   0                        50                       100

     1     463     42.9    **************************************************
     2     102      9.4    ***********
     3     210     19.4    **********************
     4      91      8.4    *********
     5     225     20.8    ************************
     6     375     34.7    ****************************************
     7     416     38.5    ********************************************
     8     139     12.9    ***************
     9     329     30.5    ***********************************
    10      47      4.4    *****
    11      94      8.7    **********
    12     153     14.2    ****************
    13      71      6.6    *******
    14     206     19.1    **********************
    15     398     36.9    ******************************************
    16     114     10.6    ************
    17      88      8.1    *********
    18     119     11.0    ************
    19      79      7.3    ********
    20      43      4.0    ****
    21      95      8.8    **********
    22     179     16.6    *******************
    23      31      2.9    ***
    24      82      7.6    ********
    25     169     15.6    ******************
    26      78      7.2    ********
    27     291     26.9    *******************************
    28     219     20.3    ***********************
    29     121     11.2    *************


       There were 216 ballots, max vote per item = 1080
   ** SORTED SIG ITEMS **


Item     Total    Percent          Percent of Largest Vote
Number   Votes    Of Max  0                       50                       100

    23    424      39.3   **************************************************
    24    393      36.4   **********************************************
    26    376      34.8   ********************************************
     1    336      31.1   ***************************************
     7    278      25.7   ********************************
    21    261      24.2   ******************************
    31    226      20.9   **************************
    20    218      20.2   *************************
     2    191      17.7   **********************
     4    185      17.1   *********************
     5    157      14.5   ******************
    22    156      14.4   ******************
     8    141      13.1   ****************
    12    136      12.6   ****************
    14    135      12.5   ***************
     3    126      11.7   **************
    27    123      11.4   **************
    25    118      10.9   *************
    10    115      10.6   *************
    15    113      10.5   *************
     6    101       9.4   ***********
    18     98       9.1   ***********
     9     82       7.6   *********
    11     81       7.5   *********
    32     77       7.1   *********
    29     71       6.6   ********
    17     67       6.2   *******
    28     62       5.7   *******
    13     42       3.9   ****
    33     39       3.6   ****
    16     37       3.4   ****
    19     36       3.3   ****
    30     23       2.1   **
   ** SIG ITEMS **

Item     Total    Percent          Percent of Largest Vote
Number   Votes    Of Max  0                       50                       100

     1    336      31.1   ***************************************
     2    191      17.7   **********************
     3    126      11.7   **************
     4    185      17.1   *********************
     5    157      14.5   ******************
     6    101       9.4   ***********
     7    278      25.7   ********************************
     8    141      13.1   ****************
     9     82       7.6   *********
    10    115      10.6   *************
    11     81       7.5   *********
    12    136      12.6   ****************
    13     42       3.9   ****
    14    135      12.5   ***************
    15    113      10.5   *************
    16     37       3.4   ****
    17     67       6.2   *******
    18     98       9.1   ***********
    19     36       3.3   ****
    20    218      20.2   *************************
    21    261      24.2   ******************************
    22    156      14.4   ******************
    23    424      39.3   **************************************************
    24    393      36.4   **********************************************
    25    118      10.9   *************
    26    376      34.8   ********************************************
    27    123      11.4   **************
    28     62       5.7   *******
    29     71       6.6   ********
    30     23       2.1   **
    31    226      20.9   **************************
    32     77       7.1   *********
    33     39       3.6   ****
```

```
     There were 216 ballots, max vote per item = 1080
```

"DEC/TEST MANAGER"

I. WHAT IS DEC/TEST MANAGER?

DIGITAL is developing a new software tool, called DEC/TEST MANAGER, to help users test their software during development and maintenance. This tool automates the organization, execution, and review of tests by several developers.

DEC/TEST MANAGER is based on the concept of regression testing. In standard regression testing, established software tests are run and the results are compared against some expected results. If the actual and the expected results do not agree, the test is considered to have failed, indicating that the software being tested may contain errors. In that case, the software is said to have "regressed."

With DEC/TEST MANAGER, you can describe your tests, classify them by assigning them to groups, and choose combinations of tests and groups to be run. DEC/TEST MANAGER executes the tests you select and compares the test results with the expected results you gave it. Multiple developers may choose to run different combinations of tests, and of course, you can always run all the tests in the test system. During the execution of a test, DEC/TEST MANAGER provides a summary of the test's status. It also allows you to view test results interactively, evaluate the test run, and use the results to make modifications to your code.

Thus, DEC/TEST MANAGER provides software developers the means to build a common test system for their project. It automates the organization of the testing, the running of the tests, and the evaluation of the results.

II. WHY WOULD I USE IT?

Testing is a necessary part of software development, yet developers frequently don't take the time or trouble to do it as well or as consistently as they should. Without adequate testing and good testing procedures, code written by several developers may not integrate smoothly. Fully tested code can save developers' time and produce a high-quality product with less expense.

With DEC/TEST MANAGER, developers must still write their own tests, but they can let the tool keep track of where those tests are stored. DEC/TEST MANAGER can run tests independently, or it can run groups of tests that share certain attributes or behaviors. The capability of grouping related tests makes it easy for developers to test modules as they finish implementing them. Because all developers have easy access to the same test system, one developer can group his tests with the related tests of another developer or with selected tests from several developers. For example, periodic integration checks of the developing software, perhaps nightly or weekly, can be performed by running all tests or significant groups of tests.

DEC/TEST MANAGER can also help with software maintenance. Once you have fixed a bug in your software, you can run a single test of that fix. You can also run that test with a group of tests that are somehow related to the code you modified. Or you can run all the tests in your test system to insure that your fix does not affect the rest of the code. DEC/TEST MANAGER gives you a consistent and simple method of testing during both development and maintenance cycles.

III. HOW DOES IT WORK?

DEC/TEST MANAGER stores all the information it needs to manage a test system in an area called a TEST MANAGER library. You create a library to hold the tests for one project or for a set of projects.

A. ORGANIZES YOUR TESTS

Once you have written your tests, you use DEC/TEST MANAGER to create a test description for each test. A test description contains the information DEC/TEST MANAGER needs to process tests, such as:

- the name of the test
- a pointer to a command procedure that runs the test
- a pointer to a set-up file that will be run before the test is run
- a pointer to a clean-up or filter file that will be run after the test is run
- a pointer to a benchmark file that contains the results you expect from the test run
- a description of what the test does

For a very simple test, not all this information is necessary. But with a complete test description, DEC/TEST MANAGER can manage a very elaborate test system.

With the flexibility to manage individual tests, simple groups of tests, and elaborate sets of groups that contain other groups or test descriptions, DEC/TEST MANAGER can keep your tests organized and easily available to all users of the test system.

B. RUNS YOUR TESTS

When you are ready to run your tests, you can collect any combination of tests and groups of tests for DEC/TEST MANAGER to execute as a single test run. By automating the process, DEC/TEST MANAGER saves developers' time but still provides a simple means for running only the tests you select. DEC/TEST MANAGER automatically compares the results of every test run with the benchmarks you supplied and generates a file of the differences and any other important information from the run.

C. AUTOMATES REVIEW OF THE RESULTS

DEC/TEST MANAGER allows you to view interactively the results of a test run and the differences recorded for each test in the run. DEC/TEST MANAGER lets you easily see which tests succeeded (that is, the actual results agreed with the benchmarks) and which tests failed (that is, the actual results differed from the benchmarks. If a test failed, you can examine the differences file produced by DEC/TEST MANAGER to determine the reason. Using the information that DEC/TEST MANAGER supplies, you can quickly organize your own work (changing and recompiling code) and then rerun the test to insure that the problem has been corrected.

DEC/TEST MANAGER helps you as you implement assigned tasks, integrate your code, and maintain your software. This tool is currently used internally by DIGITAL software developers.

For further information, contact:

Lee Rodabaugh ZK02-3/Q08
110 Spit Brook Road
Nashua, NH 03062-2897
(603) 881-2254

CONVERTING TO PDP-11 PASCAL

The following list contains syntax differences that exist between PDP-11 PASCAL and other Pascal implementations. These differences should be noted when converting your existing Pascal applications programs to run successfully under PDP-11 PASCAL. The list is not meant to be exhaustive but does point out most of the differences. For more information on the exact syntax required by PDP-11 PASCAL and the use of these features, refer to the appropriate section of the PDP-11 PASCAL Language Reference Manual or the PDP-11 PASCAL User's Guide. For information on the specific differences between PDP-11 PASCAL and VAX PASCAL refer tc Appendix E of the PDP-11 PASCAL Language Reference Manual. In PDP-11 PASCAL,

o The command line used to invoke the compiler and the command line switches may differ from those of other implementations.

o No switches embedded in comments are recognized.

o PROGRAM or MODULE headings are required. These headings must include the names of all external files which are referenced in the program or module including the standard files INPUT and OUTPUT.

o MODULEs must end with an "END." .

o %INCLUDE syntax may be different from other implementations. File names used in %INCLUDE directives must include the file extension. No default file extension is supplied. In a declaration section, the %INCLUDE directive must not be followed by a semicolon. Default file extensions are provided for files used in OPEN calls· .DAT is attached to external files, .TMP is attached to internal files.

o Only a file variable parameter is allowed for RESET and REWRITE calls. Use the PDP-11 PASCAL OPEN statement to attach specific characteristics to a file. Because the OPEN call leaves the file in an undefined state, the OPEN must be followed by a RESET or REWRITE call.

o The FIND and UPDATE predeclared procedures provide the capabilities offered by the SEEK procedure in other implementations. Calls to SEEK can be replaced by calls to FIND; the SEEK-PUT sequence can be replaced by the FIND-UPDATE sequence.

o Octal output and negative field widths are not allowed with WRITE or WRITELN statements.

o Boolean operators (AND, OR and NOT) cannot be used with integer operands.

- Octal, binary and hexadecimal constant syntax may be different from other implementations.

- The number and types of predeclared routines may be different from other implementations. Consult Appendix C of the PDP-11 PASCAL Language Reference Manual for a list of the predeclared procedures and functions provided by PDP-11 PASCAL.

- The MOD operator returns the modulus of the two operands· the REM operator returns the remainder of the division of the two operands. In some implementations, occurrences of MOD may need to be replaced by REM to achieve the same functionality.

- A new TIME function returns the time as an 11-character string. The optional OTS module FTIME returns a floating-point representation of the time.

- Declaring procedures and functions with the [EXTERNAL] attribute and/or the EXTERNAL directive causes the routine name to be passed to the Task Builder as a global symbol, even if the procedure or function is not otherwise referenced within the compilation unit.

- It is allowable to declare a routine EXTERNAL in a compilation unit and then include the body of the routine in the same compilation unit with the following restrictions. The EXTERNAL declaration must appear first with a complete parameter list and function result type, if applicable. The subsequent GLOBAL declaration must include the [GLOBAL] attribute in the routine heading and if the parameter list or function result type are included on the [GLOBAL] declaration, they will be ignored and a warning level compile-time error will be given. Usually, this warning level error will not prevent the correct operation of the program.

- Only positional parameters are allowed in calls to predeclared and user declared routines.

- REAL variables are handled with single precision only; double precision reals are not supported.

- Function result types must be simple types; structured types such as arrays and records cannot be used as function results.

- The SEQ11 directive is used to generate a FORTRAN-like calling sequence; other implementations may use a different directive for this purpose.

Following is a list of sessions, in no particular order, sponsored by the LTSIG for the Anaheim symposium:

SCIENTIFIC APPLICATION & APPLICATIONS DATABASE FOR THE PUBLIC DOMAIN. Mike Peterson, Digital Equipment Corp.

SOFTWARE TOOLS TUTORIAL & UPDATE. Dave Martin, Hughes Aircraft Company.

A COMPARISON OF SEVERAL POPULAR TEXT EDITORS. Dave Martin, Hughes Aircraft Company.

GENERIC FMS SCREEN PROCESSOR. Kirk Harmon, Varian Associates.

THE USES OF TOOLS TO MIGRATE AND CONTROL CODE FROM VMS TO UNIX Daniel Detterman, Mass. Computer Assoc.

LANGUAGES & TOOLS ROADMAP. Katherine Hornbach, Lear Siegler/Instrument Division.

INFORMAL LIBRARY CONTROL SYSTEM. Robert DeWolf, Hughes Aircraft Company.

ADVANCED OBJECT LIBRARY CONTROL TECHNIQUES. Robert DeWolf, Hughes Aircraft Company.

SORT/MERGE V3.0 PDP-11 System Software Group, Digital Equipment Corporation.

CALLING PDP-11 SORT/MERGE V3.0 FROM HIGH LEVEL LANGUAGES. PDP-11 Languages Group, Digital Equipment Corporation.

PRACTICAL BENEFITS OF A STANDARD DEVELOPMENT ENVIRONMENT ON DEC OPERATING SYSTEMS. Collins Hemingway, Oregon Software.

DEBUGGING FORTRAN-77 APPLICATIONS USING FORTRAN-77 DEBUG. Fortran Development Group, Digital Equipment Corporation.

CAPABILITIES OF DEC/TEST MANAGER -- REGRESSION TEST TOOL FOR SOFTWARE DEVELOPMENT AND MAINTENANCE. Software Tools Group, Digital Equipment Corporation.

VAX PASCAL FUTURES. Joel Clinkenbeard, Digital Equipment Corporation.

VAX PASCAL OPTIMIZATIONS. Joel Clinkenbeard, Digital Equipment Corporation.

VAX DEBUGGER V4.0. Bert Beander, Digital Equipment Corporation.

ANALYZING PROGRAM PERFORMANCE IN THE VMS ENVIRONMENT. Bert Beander, Digital Equipment Corporation.

VAX ADA(R) TECHNICAL SESSION. Charles Mitchell, Digital Equipment Corporation.

LANGUAGE SENSITIVE EDITORS. Glenn Lupton, Digital Equipment Corporation.

VAX C AND THE VMS PROGRAMMING ENVIRONMENT. Chip Nylander, Digital Equipment Corporation.

OVERVIEW OF DIGITAL'S ADA(R) LANGUAGES IMPLEMENTATION FOR VAX/VMS. Charlie Mitchell, Digital Equipment Corporation.

UNIX EMULATION USING THE VAX C RUN TIME LIBRARY. Chip Nylander, Digital Equipment Corporation.

PROGRAM EDITING IN THE VMS ENVIRONMENT. Glenn Lupton, Digital Equipment Corporation.

VAX PL/1 OVERVIEW. Chip Nylander, Digital Equipment Corporation.

A NEW EMPIRICAL ANALYSIS OF FORTRAN PROGRAMS. Joel Clinkenbeard, Digital Equipment Corporation.

FORTRAN IN THE VAX PROGRAMMING ENVIRONMENT. Joel Clinkenbeard, Digital Equipment Corporation.

DEC/CMS, DEC/MMS, DEC/TEST MANAGER -- A TUTORIAL ON VMS TOOLS. Software Tools Group, Digital Equipment Corporation.

WHAT'S NEW WITH DEC/CMS? CAPABILITES AND FEATURES. Software Tools Group, Digital Equipment Corporation.

HOW TO GET THE MOST OUT OF DEC/MMS. Software Tools Group, Digital Equipment Corporation.

VAX TPU TUTORIAL. Steve Long, Digital Equipment Corporation.

ANNOUNCING THE NEW VMS TEXT EDITOR - TPU. Steve Long, Digital Equipment Corporation.

A STANDARD FOR SOFTWARE VERIFICATION PLANS. Jay Wiley, Bechtel Power Corporation.

A STATIC ANALYZER FOR SETS OF FORTRAN MODULES. Gerald Berns, Science Applications, Inc.

HUGHES AIRCRAFTS' SOFTWARE ENGINEERING ENVIRONMENT. Joe Bryant, Hughes Aircraft Company/Ground Systems Group.

A LANGUAGE FOR PROCESSING TEXT ON VMS. Corporate Languages & Tools, Digital Equipment Corporation.

REAL WORLD USE OF SOFTWARE DEVELOPMENT TOOLS. Katherine Hornbach, Lear Siegler/Instrument Division.

PROMOTING THE ACQUISITION AND USE OF SOFTWARE DEVELOPMENT TOOLS. Katherine Hornbach, Lear Siegler/Instrument Division.

LANGUAGES AND TOOLS QUESTION AND ANSWER. Katherine Hornbach, Lear Siegler/Instrument Division.

PASCAL PROGRAM DEVELOPMENT REQUIRING ACCESS TO RSX11M EXECUTIVE DIRECTIVES. Bruce R. Ingersoll, GTE Communications Systems.

A DOD ADA STATUS REPORT. Peter Beck, U.S. Army Armament Research & Development Center.

ADA LANGUAGE SYSTEM DEMONSTRATION. Rich Thall, Softech.

MANAGING ADA IN A LARGE SYSTEM DEVELOPMENT. Mike Ryer, Intere-trics.

ADA Q&A PANEL. Peter Beck, U.S. Army Armament & Research Development Center.

AN ARTIFICIAL INTELLIGENCE PROJECT CASE STUDY -- THE FIRST SIX MONTHS. Don Rosenthal, Space Telescope Science Institute.

TEX: TYPESETTING FOR ALMOST EVERYBODY. Samuel Whidden, American Mathematical Society.

WRITING EFFICIENT AND EASY-TO-READ CODE IN VAX C. Russ Brill, Jet Propulsion Laboratory.

MULTI-LINGUAL APPLICATIONS DEVELOPMENT. Bob L. Besner, Depart-ment of National Defense.

MULTI-LINGUAL COMMERCIAL APPLICATIONS WITH VAX-11 COBOL. Mark Gillis, Digital Equipment Corporation.

MACHINE PROCESS DEFINITION SYSTEM. David W. Cohn, General Elec-tric Co.

THE USE OF DEC/CMS & DEC/MMS IN CONFIGURATION MANAGEMENT. Earl S. Cory, Eaton Corporation, Information Systems Division.

USING DCL AS A SOFTWARE DEVELOPMENT TOOL. Earl S. Cory, Eaton Corporation, Information Systems Division.

CONVERSION AND COMPARISON OF STANDARD FORTRAN, DEC FORTRAN'S AND VAX-11 FORTRAN. Earl S. Cory, Eaton Corporation, Information Systems Division.

Ada[r] Certification and Validation:

First Steps to A Production Ada System

Ronald F. Brender
Digital Equipment Corporation

June 1984

[r] Ada is a registered trademark of the U.S. Government, Ada Joint Program Office

A Super Brief History of Ada

- Early 70's - DoD spending billions on software, using hundreds of languages and dialects
- 1975 - A High-Order Language Working Group (HOLWG) established
- 1977 - DoD established that a single language to satisfy DoD requirements was feasible, but no existing language sufficed
- 1979 - The "green" language designed by Jean Iehbieh became Ada
- 1983 - ANSI/MIL-STD-1815A-1983 approved

Why Ada?

The DoD sought -

A language suited for programming DoD embedded systems, with particular concern for:

. Program reliability and maintenance

. Programming as a human activity

. Efficiency

Why Ada?

The DoD got -

A powerful general purpose language incorporating modern programming practices with integrated facilities for:

. Strong typing

. Data abstraction

. Concurrent processing (multi-tasking)

. Separate (not independent) compilation

. Exception handling

. Generic definitions

. Machine dependent facilities

Officially Ada

- Ada Standard - ANSI/MIL-STD-1815A-1983
- Ada Trademark
- Ada Validation
- Ada Board
- ISO TC97/SC5/WG14

Ada Certification and Validation

- Administered by the Ada Validation Office, AJPO
- Required to qualify for DoD contracts
- Checks conformance with the Ada standard

Certification? Validation?

Certification -

Take thd test yourself at home in private and send a note claiming you pass

Validation -

AVO comes to your house and gives you the test to see if you really pass

Certification

A letter sent by an implementer to the AVO that describes:

- All host and/or target configurations to be validated
- All implementation specific parameters (range of integers, accuracy of floating point, and the like)
- All implementation dependent characteristics according to the criteria in Appendix F of the Ada Standard

Certification

Also, a statement that:

- o The implementation passes all applicable tests
- o No deliberate extensions to the Ada Standard exist in the implementation

And agreement to:

- o Public release of all information concerning the results of validation testing
- o Comply with the Ada Trademark Policy of the AJPO

Certification

Further, a description of and justification for:

- o Any modifications required to pass any test
- o Any tests claimed to be incorrect
- o Any tests deemed not applicable
- o Compiler options available and used

Certification

Finally, assure the availability during actual validation of:

- o Complete hard copy listings of all tests
- o Access to computing facilities on a 24-hour basis
- o Machine readable output to be kept by the AVO (if at all possible)

Certification Acceptance

The letter of Certification is accepted when:

- o The AVO is satisfied that all required information is provided
- o Any disputed tests have been reviewed by a "Fast Reaction Team" of Ada experts
  - . The AVO withdraws any disputed test that is resolved in favor of the implementer
  - . The implementer agrees to conform to any disputed test resolved in favor of the AVO

Actual validation is then scheduled

Validation

- AVO personnel come on-site and supervise running of the tests
- If 100 percent of the applicable tests are passed then the implementor receives a

  Certificate of Validation
- A certificate of validation is good for one year

The Ada Validation Suite - "The Test"

- A set of about 1800 tests
- Checks that correct programs work
- Checks that incorrect programs are diagnosed
- Augmented, revised and updated on a six-month cycle
- Version 1.4 as of April 1984
- Publicly available

Kinds of Validation Tests

A*	Executable, no internal checking
B*	Non-executable, deliberate errors that must be detected during compilation
C*	Executable, self-checking with automatic reporting
D*	Capacity tests (do not influence success of validation)
E*	Executable, determine certain options, check non-binding interpretations
L*	Non-executable, deliberate errors which must be detected no later than linking

Kinds of Validation Tests

*.ADA	Normal tests
*.DEP	Implementation dependent tests For options like LONG_INTEGER, SHORT_FLOAT data types
*.TST	Parameterized tests Must adapt for implementation dependent parameters, such as maximum digits of floating point precision

Distribution of Tests (Units)

	*.ADA	*.DEP	*.TST
A*	57	1	1
B*	755	9	21
C*	918	340	10
D*	14	-	-
E*	6	-	1
L*	46	14	-

Distribution of Tests (Units)

Tests	Chapter	
-	1	Introduction
92	2	Lexical Elements
291	3	Declarations and Types
363	4	Names and Expressions
233	5	Statements
105	6	Subprograms
77	7	Packages
164	8	Visibility Rules
179	9	Tasks
219	10	Program Structure ...
26	11	Exceptions
214	12	Generics
-	13	Representation Clauses ...
222	14	Input-Output

Digital Status on 7 June 1984

Per cent of the validation tests passed:	???	%
Disputed tests:	???	
Non-applicable tests:	???	
Letter of certification submitted:	???	1984
Validation scheduled:	???	1984

Significance of Validation

Being validated means that the implementation

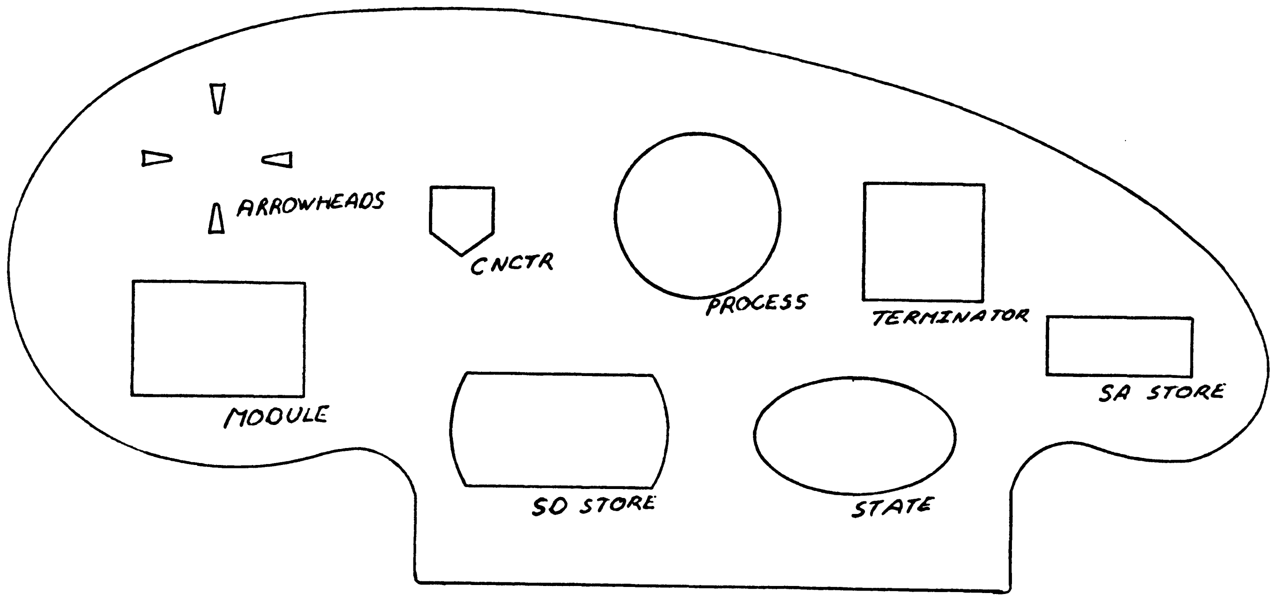
- o Is able to pass a reasonably demanding set of tests
- o Qualifies for consideration where "a validated compiler" is a selection criteria

No More, No Less

<u>Languages and Tools "Mushroom"</u>

Keeping in spirit with the Languages and Tools SIG's emphasis on software development <u>methods</u>, as well as languages and tools, the SIG had designed and manufactured a special template for sale in the DECUS store at Cincinnati and later Symposia. The template contains all the symbols necessary for the drawing of <u>data flow diagrams</u> and <u>structure charts</u>, which are the graphical outputs from the Structured Analysis/Structured Design methodology. It even includes the French curves necessary for drawing the data flows -- and hence its unusual, mushroom-like outline.

Those of you who have tried to do SA/SD by hand before will realize that it takes at least three commercially-available templates to provide all the symbols needed. The L & T template provides them all...you can see its outline below.



The LT SIG Standards Activities

My name is Jay W. Wiley and I am the Standards Coordinator for the LT SIG. I can be reached at:

Bechtel Power Corporation
12400 E. Imperial Highway
Norwalk, CA 90650
213-807-4016

The standards activities that are of interest to the SIG are the national standards associated with the programming languages that are represented by the SIG and software engineering standards. The language specific standards activities are normally handled by the SIG language coordinator. Software engineering standards typically fall into two categories. These categories are Department of Defense (DOD) and non-DOD. The DOD standards are of primary interest to military contractors and will not be addressed by the SIG. The most active non-DOD group is the Software Engineering Standards Subcommittee (SESS) of the IEEE Computer Society. I am an active member of Working Group P1012, a Standard for Software Verification Plans, and will present a paper on the activities of this Working Group during the Fall Symposium. Included in this newsletter is a copy of the current SESS status report. This report includes the names and addresses of the Chairmen of all of the various Working Groups. Anyone can participate in these working groups. You do not need to be a member of IEEE.

The SIG will have information on the proposed FORTRAN 8X standard available at the Fall Symposium. If you are planning on attending the Symposium, be sure to stop in at the LT SIG suite.

STATUS REPORT, 1 May 1984

Software Engineering Standards Subcommittee

Technical Committee on Software Engineering

IEEE Computer Society

SESS Status Report, 1 May 1984

TABLE OF CONTENTS

1.0 PURPOSE 1
2.0 SUMMARY OF THIS YEAR'S ACTIVITY TO DATE 1
2.1 Completed Events 1
2.2 New Starts 1
3.0 APPROVED STANDARDS 2
3.1 ANSI/IEEE 729-1983. IEEE Standard Glossary Of Software Engineering Terminology 3
3.2 ANSI/IEEE 730-1981, IEEE Standard For Software Quality Assurance Plans 3
3.3 IEEE Std 828-1983, IEEE Standard For Software Configuration Management Plans 4
3.4 ANSI/IEEE 829-1983, IEEE Standard For Software Test Documentation 4
3.5 IEEE Std 830-1983, IEEE Guide For Software Requirements Specifications 4
4.0 APPROVED PROJECTS 5
4.1 P730-1, A Standard For Software Quality Assurance Plans 6
4.2 P982, A Standard For Software Reliability Measurement 7
4.3 P983, A Guide For Software Quality Assurance . . . 8
4.4 P990, A Guide For The Use Of Ada* As A Program Design Language 8
4.5 P1002, Software Engineering Standards Taxonomy . . 9
4.6 P1008 A Standard For Software Unit Testing . . . 11
4.7 P1012 A Standard For Software Verification Plans 12
4.8 P1016 A Guide For Software Design Descriptions 13
4.9 P1028, A Standard For Software Reviews And Audits 13
5.0 TENTATIVE PROJECTS 14
5.1 A Guide For Software Configuration Management Plans 15
5.2 A Standard Classification Of Software Errors, Faults, And Failures 16
5.3 A Standard For Software Productivity Metrics . . 17
5.4 A Guide For Third Party Software Acquisition . . 17
5.5 A Standard For Software Quality Metrics 18
5.6 A Standard For User Documentation 18
6.0 SOFTWARE ENGINEERING STANDARDS SEMINARS 19
7.0 SOFTWARE ENGINEERING STANDARDS APPLICATION WORKSHOPS 20
8.0 OTHER ITEMS 20
8.1 Organizational Representation 20
8.2 Organizational Points Of Contact 22
8.3 Other Standards Activities Of Interest 23
8.3.1 ISO/TC-97 23
8.3.2 P610, Computer Dictionary 23
8.3.3 ANS 10.4 Guideline For Software Verification And Validation 24

8.3.4 ASTM E 730 24
8.4 SESS Administration 24
8.4.1 SESS Guide 25
8.4.2 Long-Range Planning 25
8.5 Miscellaneous 25

---

* Ada is a registered trademark of the United States Government Department of Defense (AJPO).

SESS Status Report, 1 May 1984

1.0 PURPOSE

The purpose of this memo is to provide the status of efforts on:

1. Approved standards.
2. Approved projects.
3. Tentative projects.
4. Software Engineering Seminars
5. Other items.

2.0 SUMMARY OF THIS YEAR'S ACTIVITY TO DATE

Significant events occuring to date in 1984 to date are summarized as follows:

2.1 Completed Events

1. The IEEE Standards Board published IEEE Std 830-1984, IEEE Guide to Software Requirements Specifications, on 10 Feb 1984.
2. The 1985 SESS Budget request was provided to J. Musa, TCSE Chairperson, and to T. Pittman, CSC Chairperson, on 28 Feb 1985 by S. Gloss-Soler
3. The Charter and Organization of the SESS was composed by A. Frank Ackerman, reviewed and revised by a group chaired by Shirley Gloss-Soler, and submitted to a ballot of the SESS Executive Board on 20 March 1984.
4. The Software Engineering Techniques project has been cancelled

2.2 New Starts

1. The following new projects were initiated on 28 Feb 1984

a. A Standard for Software Quality Metrics

b. A Standard for User Documentation

c. A Guide for Third Party Software Acquisition

2. The Project Authorization Request for A Standard for Software Reviews and Audits was approved by the IEEE Standards Office at the 22 March 1984 Standards Board meeting. This was subject to a change in the name of the project to "Guide" or "Recommended Practice". That directed change is currently under appeal to the Standards Board

3.0 APPROVED STANDARDS.

(Copies of these Standards may be purchased using the order form at the end of this report.)

These are summarized in Table 1.

Table 1
Approved Software Engineering Standards

ANSI/IEEE Std 730-1981,	IEEE Standard for Software Quality Assurance Plans
ANSI/IEEE Std 729-1983,	IEEE Standard Glossary of Software Engineering Terminology
ANSI/IEEE Std 829-1983,	IEEE Standard for Software Test Documentation
IEEE Std 828-1983,	IEEE Standard for Software Configuration Management Plans
IEEE Std 830-1984,	IEEE Guide to Software Requirements Specifications

3.1 ANSI/IEEE 729-1983. IEEE Standard Glossary Of Software Engineering Terminology

1. Chairperson is:

   S. Gloss-Soler
   (315) 456-1240
   1607 Craig St
   Rome, NY 13440

2. The standard was approved by ANSI on 5 Aug 1983

3. Further action on the International area is being held in abeyance pending completion of the efforts on the establishment of ISO/TC-97/SC-22. (See 8.3.1)

3.2 ANSI/IEEE 730-1981, IEEE Standard For Software Quality Assurance Plans.

1. Chairperson is:

   Fletcher J. Buckley
   (609) 778-3606
   RCA, Mail Stop 101-229
   Moorestown, NJ 08057

2. Status is as follows; the Standard was

   a. Approved by ANSI on 21 July 1982.

   b. Submitted to the Secretariat, ISO/TC-176 by the Chairman, USTAG on 17 March 1983.

   c. Submitted to the Chief, US Delegate, ISO/TC-97 on 24 Feb 1983.

3. Projected milestones are as follows:

   a. Approved by ISO/TC-176 at their next meeting: March 1984.

   b. Approved by ISO/TC-97: March 1984.

3.3 IEEE Std 828-1983, IEEE Standard For Software Configuration Management Plans.

1. Chairperson is:

   Rick Fredrick
   (214) 995-2690
   Texas Instruments, M. S. 8
   PO Box 5012
   Dallas, Texas 75067

2. The standard was approved by the IEEE Standards Board on 24 June 1983.

3. The results of the meeting with NPEC were approved by the IEEE Standards Board on 1 Feb 1984.

4.

3.4 ANSI/IEEE 829-1983, IEEE Standard For Software Test Documentation

1. Chairperson is:

   David Gelperin
   (612) 542-8620
   2425 Zealand Ave, N.
   Golden Valley, Minn. 55427

2. This standard was approved by ANSI on 19 August 1983.

3. Further action on the International area is being held in abeyance pending completion of the efforts on the establishment of ISO/TC-97/SC-22. (See 8.3.1)

3.5 IEEE Std 830-1983, IEEE Guide For Software Requirements Specifications

1. Chairperson is:

   Al Davis
   (602) 582-7069
   GTE Network Systems

2500 West Utopia Road
Phoenix, Az 85027

2. The Guide was approved by the IEEE Standards Board on 22 Sept 1983, and published on 5 Feb 19984.

3. Projected milestone is as follows:

   a. To be adopted by ANSI: July 1984

4.0 APPROVED PROJECTS

These are shown in Table 2.

Table 2
Approved Software Engineering Standards Projects

A Standard for Software Quality Assurance Plans (Revision) (P730-1)

A Standard for Software Reliability Measurement (P982)

A Guide for Software Quality Assurance (P983)

A Guide for the Use of Ada* as a PDL (P990)

Software Engineering Standards Taxonomy (P1002)

A Standard for Software Unit Testing (P1008)

A Standard for Software Verification Plans (P1012)

A Guide for Software Design Descriptions (P1016)

A Standard for Software Reviews and Audits (P1028)

*Ada is a registered trademark of the US Government, AJPO.

4.1 P730-1, A Standard For Software Quality Assurance Plans.

1. Chairperson is:

   Fletcher J. Buckley
   (609) 778-3606
   RCA, Mail Stop 101-229
   Moorestown, NJ 08057

2. Co-Chairperson is:

   Robert Felker
   (215) 770-6675
   Penn Power and Light
   2 North 9th St
   Allentown, Penn 18101

3. Current status is as follows

   a. The balloting has been completed. All members of the balloting group and coordination groups have been notified of the results and offered an opportiunity to change their ballot.

   b. Results of the balloting effort are as follows:

      (1) Number of ballots sent. 181

      (2) Number of ballots returned: 154 (85.1%)

      (3) Affirmatives: 144 (93.5% of returns)

      (4) Number of unresolved Negative Votes: 1 (0.6% of returns)

      (5) Number of Abstentions: 9 (5.9% of returns)

4. The revised standard was provided to the IEEE Standards Board on 15 March 1984 for approval at their June 1984 meeting.

4.2 P982, A Standard For Software Reliability Measurement

1. Chairperson is:

   Jim Dobbins
   (703) 367-3912
   MS 105-9013
   IBM FSD
   9500 Godwin Drive
   Manassas, Va 22110

2. Co-chairpersons are:

   a. Ted Workman
      011 44 344 77319, ext 3647
      Hewlett-Packard Limited
      Nine Mile Ride
      Wokingham, Berks
      England RG 11 3LL

   b. Ray Leber
      (215) 962-4118
      General Electric Co.
      PO Box 8555, Room M2216, Bldg 100
      Philadelphia, Pa 19101

3. The last meeting was held on 2-4 April 1984: Orlando, Fla.

4. The schedule for forthcoming meetings is as follows:

   a. October 1984, San Francisco, Calif.

5. Status is as follows:

   A initial final draft has been produced.

6. Projected milestone is as follows:

   Draft standard balloted: Jan 1985

4.3 P983, A Guide For Software Quality Assurance.

1. Chairperson is:

   G. Tice
   (503) 629-1310
   Tektronix
   PO Box 392
   Wilsonville, Oregon 97070

2. Co-chairperson is:

   A.K. Ackerman
   (201) 981-7946
   Bell Laboratories,
   8 Corporate Place
   Piscataway, NJ 08854

3. The last meeting was held at Dallas Texas: 8-10 May 1984.

4. The schedule of forthcoming meetings is as follows:

   a. Portland, Oregon: 28-30 Aug 1984

5. Status is as follows:

   a. A Project Authorization Request was approved by the IEEE Standards Board at their December 1982 meeting with the addition of coordination with ISO/TC 97.

   b. The first complete draft was produced and distributed for comment on 2 Aug 1983.

6. Projected milestone is as follows:

   Draft standard balloted: Sept 1984

4.4 P990, A Guide For The Use Of Ada* As A Program Design Language

---

* Ada is a registered trademark of the United States Government Dept of Defense (AJPO).

SESS Status Report, 1 May 1984

1. The chairperson is:

   Bob Blasewitz
   (609) 778-3955
   RCA, MS 101-210
   Moorestown, NJ 08057

2. The co-chairperson is:

   Mark S. Gerhardt
   (401) 847-8000
   Raytheon
   PO Box 360
   Portsmouth, RI 02871

3. The last meeting was held on 24-26 April 1984, Atlanta, Ga

4. The schedule of forthcoming meetings is as follows:

   a. TBD

5. Status is as follows:

   a. The US Government Dept of Defense Ada Joint Program Office (AJPO) provided permission to this organization to use their trademarked term "Ada", in connection with this project, by telegram on 11 March 1983.

   b. A Project Authorization Request was approved by the IEEE Standards Board at their 17 March 1983 meeting.

   c. The scope and outline have been completed.

6. Projected milestones are as follows:

   a. First complete draft: Dec 1983.

   b. Draft standard balloted: Sept 1984.

4.5 P1002, Software Engineering Standards Taxonomy

1. Chairperson is:

   Leonard Tripp

(206) 575-5390
Boeing Computer Services
MS 9C-24, PO Box 24346
Seattle, Washington, 98124

2. Co-Chairpersons are:

   a. Ralph Wachter
      (301) 953-7100, Ext 7336
      Johns Hopkins APL
      11100 Johns Hopkins Road
      Laurel, Maryland 20707

   b. Perry Nuhn
      (203) 375-0200 ext 266
      ITT Programming Technology Center
      1000 Oronoque Lane
      Stratford, Conn 06443

3. The last meeting was held on 23-25 April 1984: St. Louis, Missouri.

4. The schedule of forthcoming meetings is as follows:

   a. June/July: Washington, DC.

5. Status is as follows:

   The Project Authorization Request was approved by the IEEE Standards Board on 24 June 1983.

   b. Four items have been released for comment to interested parties:

      (1) A Project Plan, dated 28 March 1983

      (2) A Statement of Requirements, dated 28 March 1983

      (3) A bulletin requesting involvement in clarifying some fundamental issues in defining a Software Engineering Standards Taxonomy.

      (4) A bulletin requesting support in building a Taxonomy Reference Library.

6. Projected milestones are as follows:

   a. First complete draft: Sept 1984.

   b. Draft standard balloted: April 1985.

4.6 P1008 A Standard For Software Unit Testing

1. Chairperson is:

   David Gelperin
   (612) 541-1431
   Software Quality Engineering
   2425 Zealand Ave, N.
   Golden Valley, Minn 55427

2. Co-chairpersons are:

   a. Hugh Spillane
      (617) 299-3801
      RCA, A/S, Mail Stop 18-2
      PO Box 588
      Burlington, Mass 01803

   b. Pat Wilburn
      (509) 376-4711
      Westinghouse - Hanford Co.
      M/S W/B-109
      PO Box 1970
      Richland, Washington 99352

3. The last meeting was 8-11 May 1984, Toronto, Canada

4. Schedule of forthcoming meetings is as follows:

   a. 7-10 Aug 1984, Seattle, Washington.

   b. 13-16 Nov 1984, Atlanta, GA.

5. The Project Authorization Request was approved by the IEEE Standards Board on 23 June 1983 The Scope and outline have been completed.

6. Projected milestones are as follows:

   a. First complete draft: May 1984.

   b. Draft standard balloted: June 1985.

4.7 P1012 A Standard For Software Verification Plans

1. Chairperson is:

   Roger Fujii
   (213) 831-0611 Ext 2420
   Logicon, Inc.
   255 West Fifth St.
   San Pedro, Calif 90731

2. Co-Chairperson is:

   Douglas McMann
   (213) 535-4917
   TRW Defense and Space Systems Group
   Mail Stop M-1/1406
   1 Space Park
   Redondo Beach, Calif 90273

3. The last meeting was on 28 Feb -1 March 1984, San Francisco, Calif

4. Schedule of forthcoming meetings is as follows:

   a. 19-21 June 1984, New Orleans

   b. 11-13 Sept 1984 Seattle

5. The Project Authorization Request was approved by IEEE Standards Board on 22 Sept 1983, subject to coordination with ISO/TC97.

6. Contact has been established with the ANS 10.4 committee which is working on a similiar topic for the American Nuclear Society (see below).

4.8 P1016 A Guide For Software Design Descriptions

1. Chairperson is:

   H. Jack Barnard
   (303) 538-3976
   Mail Stp 1D30
   ATT, Inf. Sys. Labs
   11900 North Pecos St.
   Denver, Colorado, 80234

2. Co-chairperson is:

   Jim Darling
   (303) 673-7617
   Storage Technology Corp
   Mail Stop 93
   2270 South 88th St.
   Louisville, Co. 80028

3. The last meeting was held on 27-29 March 1984: Orlando, Fla.

4. Schedule of forthcoming meetings is as follows:

   a. July 1984, Las Vegas, Nevada

   b. September 1984, Toronto

5. The Project Authorization Request was approved by the IEEE Standards Board, on 22 Sept 1983 subject to coordination with ISO/TC97. The Standards Board recommended that the title be changed from "Guide" to "Recommended Practice".

6. The second draft is scheduled to be distributed in October 1984

4.9 P1028, A Standard For Software Reviews And Audits

1. Chairperson is:

   Charles P. Hollocker
   (312) 979-5823
   ATT
   901 Rolling Drive
   Lisle, Il 60532

2. Co-Chairperson is:

   Tim Kasse
   (602) 438-3572
   Mororola Microsystems
   2900 South Diablo Way
   Tempe, Arizona 85062

3. The last meeting was 28-29 Feb , 1 March 1984, San Francisco, Calif.

4. Further meetings are projected as follows:

   a. 5-8 June, Chicago.

   b. 18-21 Sept 1984, Arlington, Va

5. Current status is that The Project Authorization Request has been approved by the IEEE Standards Office at the 22 March 1984 Standards Board meeting. This was subject to a change in the name of the project to "Guide" or "Recommended Practice". That directed change is currently under appeal to the Standards Board

5.0 TENTATIVE PROJECTS.

These are shown in Table 3.

Table 3
Tentative Software Engineering Standards Projects

A Guide for Software Configuration Management

A Standard Classification for Software Errors,
Faults and Failures

A Standard for Software Productivity Metrics

A Standard for Software Quality Metrics

A Standard for User Documentation

A Guide for Third Party Software Acquisition

5.1 A Guide For Software Configuration Management Plans

1. Chairperson is:

   Richard Van Tilburg
   (714) 732-2307
   Hughes Aircraft Corporation
   Bldg 618, MS B209
   PO Box 3310
   Fullerton, Calif 92634

2. Co-Chairperson is:

   David Schwartz
   (602) 869-3827
   Intel Corp, MS: DZ 2-274
   2402 West Beardsley Road
   Phoenix, AZ 85027

3. Current status is that a Project Authorization Request was provided to the Secretary of the IEEE Standards Board on 18 March 1984 by the Chairperson, CSC. It is projected that this will be approved at the June meeting of the Standards Board.

4. The last meeting was held at 30-31 Jan 1984, Santa Barbara, Calif

5. Further meetings are projected as follows:

   a. 16-19 May 1984, Phoenix, Az

   b. Sept 1984, Washington, DC

   c. Jan 1985, Austin, Texas

5.2 A Standard Classification Of Software Errors, Faults, And Failures.

1. Chairperson is:

   Dick Evans
   (213) 536-3805
   TRW, Mail Stop R4/2182
   One Space Park
   Redondo Beach, California 90728

2. Co-Chairperson is:

   David Simkins
   (607) 751-5346
   IBM, FSD
   M.C. 889 B P75
   Owego, New York 13827

3. The last meeting was held on 7-8 Feb 1984, Redondo Beach, Calif.

4. Further meetings are projected as follows:

   a. 23-25 May, Toranto, Canada

   b. 25-27 Sept 1984, Bethesda, Md.

   c. Jan 1985, Los Angeles

   d. April 1985 TBD

5 Current status is that a Project Authorization Request was provided to the Secretary of the IEEE Standards Board on 18 March 1984 by the Chairperson, CSC. It is projected that this will be approved at the June meeting of the Standards Board.

5.3 A Standard For Software Productivity Metrics

1. Chairperson is:

   Eleanor Antreassian
   (201) 981-6479
   Bell Labs
   Room 3B121
   6 Corporate Place
   Piscataway, New Jersey 08854

2. Co-Chairperson is:

   Robert Sulgrove
   (513) 445-1064
   NCR Corp., WHQ-5E
   1700 South Patterson Blvd
   Dayton, Ohio 45479

3. The last meeting was held on 18-20 Jan 1984, Melborne, Fla.

4. Further meetings are projected as follows:

   a. 2-4 May, Oregon.

   b. Sept, Nashua, NH

5. Current status is that a Project Authorization Request was sent to the Chairperson of the Computer Standards Committee 1 April 1984. It is projected that this will be approved at the June meeting of the Standards Board.

5.4 A Guide For Third Party Software Acquisition

1. Chairperson is:

   Philip C. Marriott
   (513) 445-2198
   NCR Corporation
   World Hqs 4
   Dayton, Ohio 45479

2. Co-Chairperson is:

   Flo Harteloo

(602) 438-3068
Motorola Microsystems
Mail Stop DW 212
2900 South Diablo Way
Tempe, Ariz 85282

3. The first meeting was held, 28 Feb 1984, San Francisco, Calif.

5.5 A Standard For Software Quality Metrics

1. Chairperson is:

   Dr. Norman F. Schneidewind
   (408) 646-2719/3211
   Professor, Dept RSA/CS
   Code 54SS
   Naval Postgraduate School
   Monterey, Calif 93940

2. The first meeting was held, 28 Feb 1984, San Francisco, Calif.

5.6 A Standard For User Documentation

1. Chairperson is:

   Christopher Cooke
   (301) 338-5652
   Martin Marietta Aerospace
   Mail Station 98
   103 Cheasapeake Park Plaza
   Baltimore, Md 21220

2. The first meeting was held 28 Feb 1984 at the Cathedral Hotel, San Francisco, Calif.

3. The next meeting is scheduled for 11-12 June 1984, Naperville, Illinios.

6.0 SOFTWARE ENGINEERING STANDARDS SEMINARS

Table 4 shows the schedule of Software Engineering Seminars sponsored by this subcommittee jointly with the IEEE Standards Board.

Table 4
Projected Schedule of Software Engineering Seminars

Seminar Title	Date	Place
Software Quality Assurance	23-25 May 1984	San Francisco, Calif.
	4-6 July 1984	London, England
	10-12 Sept 1984	Washington, DC
	14-16 Nov 1984	San Francisco
Software Test	13-14 Sept 1984	Washington, DC
	22-23 Oct 1984	Texas
	26-27 Nov 1984	San Diego, CA
Software Configuration Management	9-10 Oct 1984	Atlantic City, NJ
	24-25 Oct 1984	Texas
	28-29 Nov 1984	San Diego, CA
Software Requirements	Spring 1985	

For details on these seminars, contact should be made with the following:

S. Havranek
212 705-7907
Seminar Marketing Manager
IEEE Standards Board
345 East 47th St
New York, New York 10017

7.0 SOFTWARE ENGINEERING STANDARDS APPLICATION WORKSHOPS

The Third Software Engineering Standards Application Workshop is scheduled for 2-4 Oct 1984 at the Sherton-Palace Hotel, San Francisco, Calif.

Chairperson is:

Leonard Tripp
(206) 575-5390
Boeing Computer Services
MS 9C-70, PO Box 24346
Seattle, Washington, 98124

8.0 OTHER ITEMS

8.1 Organizational Representation

The following are the current representatives to this subcommittee from standards-making organizations:

1. American Institute of Aeronautics and Astronautics (AIAA) Software Systems Technical Committee:

   Robert R. Jones
   (714)732-3849
   Hughes Aircraft Corp.
   PO Box 3310, MS 618/b218
   Fullerton, Calif 92634

2. ANS/NPEC Joint Working Group

   a. N. C. Farr
      (812) 289-3000 ext 1495
      Public Service Indiana
      PO Box 190
      New Washington, IN 47162

   b. Jim Thomas (alternate)
      (704) 373-4612
      422 South Church St.
      Charlotte, NC 28242

3. American Nuclear Society

   Dr Gerald W. Main
   Hanford Engineering Development Lab

Westinghouse Hanford Co.
PO Box 1970
Richland, WA 99352

4. ANSI X , IEC/TC-83, ISO/TC-97

E. Lohse
240 Radnor - Chester Road
Radnor, PA 19087

5. ANSI Z1:

John Milandin
(313) 994-7778
Bechtel Power Corp.
777 East Eisenhower Pkwy
Ann Arbor, Mich 48106

6. ASQC:

Art Ferlan
(603) 884-6711
Digital Equipment Corp.
M/S MK 1-2/G15
Continental Blvd
Merrimack, NH 03054

7. ASTM

Peter E. Schilling, Chairman
ASTM Comm E-31 (CS)
(412) 337-2724
Alum. Co. of America
Alcoa Technical Center
Alcoa Center, PA 15069

8. DPMA:

William E. Perry
(305) 876-4292
Quality Assurance Institute
9222 Bay Point Drive
Orlando, Fla 32811

9. EDP Auditors Association:

Stanley R. Jarocki
Bankers Trust Co.
Four Albany Street
New York, NY 10015

10. EIA:

    H. Ronald Berlack
    (603) 885-5170
    Sanders Associates
    M/S NCA1-4222
    95 Canal St
    Nashua, NH 03061

11. IEEE Reliability Society

    Dr. T. L. Regulinski
    (602) 932-7321
    Goodyear Aerospace
    PO Box 295
    Goodyear, Ariz 85338

12. National Security Industrial Association (NSIA)

    Perry Nuhn
    (203) 375-0200 ext 266
    ITT Programming Technology Center
    1000 Ornoque Lane
    Stratford, Conn 06443

8.2 Organizational Points Of Contact

The following are organizational points of contact:

1. ADAPSO

   Lauren Erling
   (703) 522-5055
   Suite 300
   1300 North 17th St
   Arlington, Va 22209

2. ANS X3 K1

   Michael Landes, Chairperson
   (703) 521-5280, ext 271
   ANS X-3 K1, Program Documentation
   Computer Science Corp.
   400 Army-Navy Drive
   Arlington, Va 22202

3. W.F. Michell
   01-629-9000
   British Standards Institution
   2 Park St.
   London W1A2BS, England

4. Dept of Defense

   William P. LaPlant
   (202) 695-0499
   AF/ACDT, Rm 5C1084
   The Pentagon
   Washington, DC 20330

8.3 Other Standards Activities Of Interest

The activities listed below are not a part of the effort of this Subcommittee. They are of direct interest, however, and so are identified herein.

8.3.1 ISO/TC-97 - The following information is extracted from the minutes of the IEEE Standards Board of 24 June 1983.

Secretary's Report: "At a meeting of the X3 Committee held in Washington on June 2 and 3, the Committee , acting as the Technical Advisory Group to TC-97, asked IEEE to assume responsibility as Secretariat to act as the Technical Advisory Group for a new Activity within TC-97 on software engineering. . . . "

Report of Standards Board actions: "7. Authorized IEEE to assume the Secretariat of a new ISO sub-committee (tentatively identified as ISO TC97/SC22) on Software Engineering."

8.3.2 P610, Computer Dictionary -

1. Chairperson is:

   Jane Radatz
   (619) 455-1330
   Logicon, Inc
   PO Box 80158
   San Diego, Calif 92138

2. This effort is sponsored by the IEEE Computer Society Computer Standards Committee.

3. The last meeting was 4-5 Nov 1983, at Rosslyn, Va.

4. Next meeting is scheduled for 4-5 Feb 1984, Washington, DC.

8.3.3 ANS 10.4 Guideline For Software Verification And Validation -

1. Contact is:

   Dr. Odelli Ozer
   Electric Power Research Institute
   3412 Hillview Ave
   PO Box 10412
   Palo Alto, Calif 94303

2. This effort is sponsored by the American Nuclear Society (ANS) Committee 10.4.

3. As of 1 June 1983, they were in the sixth draft of their guideline.

8.3.4 ASTM E 730 - The ASTM G-31 Committee is currently initiating a revision of ASTM Standard E 730, Guide for Functional Design of Computerized Systems.

1. Chairperson of the effort is:

   Hearn Buttrill
   (301) 757-1854
   1616 Meeting House Lane
   Annapolis, Md 21401

2. Comments are requested by 2 Dec 1983.

8.4 SESS Administration

SESS Status Report, 1 May 1984

8.4.1 SESS Guide - The SESS Guide to Standards Development was approved by ballot of the SESS Executive Board on July 1983.

8.4.2 Long-Range Planning - A Five-Year Plan for SESS Activities is in draft stage. Interested personnel should contact:

Thomas M. Kurihara
(202) 755-1771
2058 Carrhill Road
Vienna, Virginia 22180

8.5 Miscellaneous

1. The next SESS Executive Board meeting is scheduled for Fall CompCon, Arlington, Va, during the week of 17 Sept 1984.

Fletcher J. Buckley
Chairperson
Software Engineering Standards Subcommittee
TC on Software Engineering
IEEE Computer Society

DECUS

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER SOCIETY
249 NORTHBORO ROAD, (BPO2)
MARLBORO, MA 01752

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: ______________
Name: ______________
Company: ______________
Address: ______________
____
State/Country: ______________
Zip/Postal Code: ______________
Phone No.: ______________

Mail to: **DECUS - Attn: Subscription Service**
249 Northboro Road, (BPO2)
Marlboro, MA 01752 USA

Affix mailing label here. If label is not available, print old address here. Include name of installation, company, university, etc.

Bulk Rate
U.S. Postage
PAID
Permit No. 18
Leominster, MA
01453