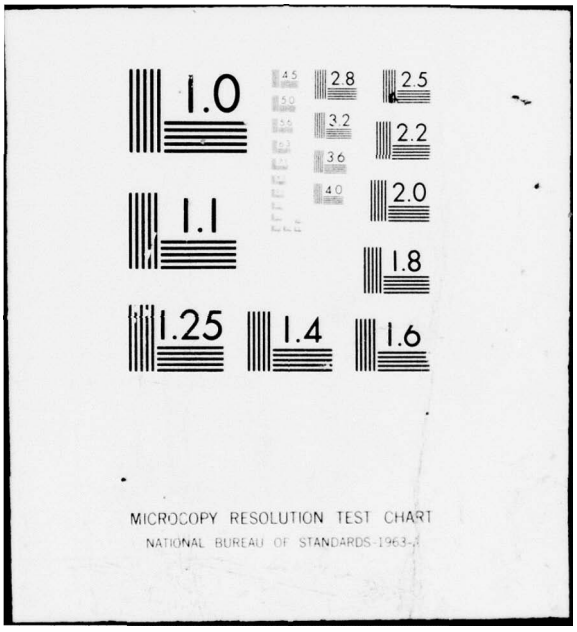
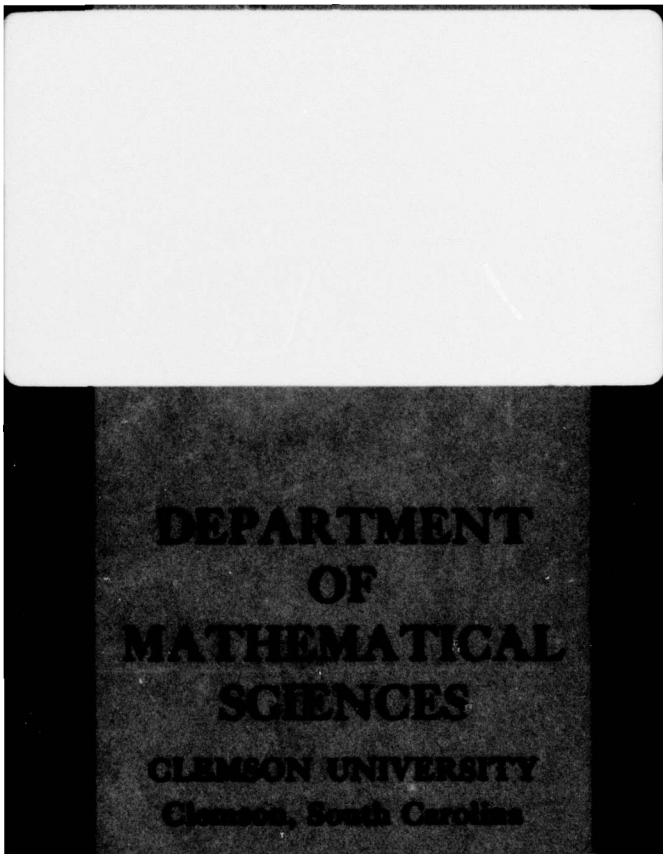
AD-A048 248 CLEMSON UNIV S C DEPT OF MATHEMATICAL SCIENCES F/G 9/2
GENERAL CONSIDERATIONS ON THE DESIGN OF AN INTERACTIVE SYSTEM F--ETC(U)
NOV 77 R F LING N00014-75-C-0451
UNCLASSIFIED N93 NL

| OF |
AD
A048 248

END
DATE
FILMED
-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


DEPARTMENT
OF
MATHEMATICAL
SCIENCES
CLEMSON UNIVERSITY
Clemson, South Carolina

(6)

GENERAL CONSIDERATIONS ON THE DESIGN OF AN INTERACTIVE SYSTEM FOR DATA ANALYSIS.

BY

(10) ROBERT F./LING

(9) Technical rept.,

INVITED PAPER: ORSA/TIMS CONFERENCE
ATLANTA GEORGIA
(11) NOVEMBER, 1977

(12) 16p.

(14) REPORT N93,
(TR-269)

(15) SUPPORTED IN PART BY THE OFFICE OF NAVAL RESEARCH
CONTRACT N00014-75-C-0451 TASK NR 042 - 271

ABSTRACT

Among the most important criteria in the design and implementation of an interactive system for data analysis are: data structure, control language and user interface, system versatility, extensibility, and portability. The design of an interactive system, viewed as a sequential consideration of these criteria, will be discussed.

1. INTRODUCTION

The commonly accepted meaning of the term "interactive" in "interactive systems" has gone through a rapid evolution in the past decade. In the early days, "interactive computing" generally meant "remote entry of batch jobs from a terminal." This was accomplished by modifying the source codes of batch programs for the input of control-card information (and data) to accept input from the terminal via a fixed sequence of prompts. Such a mode of operation gave rise to the expression "conversational program", although almost all of the "conversations" were initiated by the program and not by the user, so that there were very little genuine man-machine-interactions.

Today, the concept of interactive computing is considerably more advanced and sophisticated, so is the design and implementation of systems supporting such a mode of computing. The terms "interactive" and "terminal-oriented" (or timesharing) are no longer synonymous. I shall use "an interactive system for data analysis" to mean a system which is capable of supporting a high level of efficient, man-machine interaction in data analysis, when the analysis calls for a sequential decision procedure by the user, with conditional multiple branches at each step depending on the intermediate results of the previous steps.

A well-designed interactive system will, at the very least, enable its user to attempt one or more iterations of data editing, plotting, transformations, or new analyses, without re-initiating the system or re-entering the data values. The same analysis, using batch systems or inflexible terminal-oriented systems, will necessitate many separate runs, each of which will duplicate some steps of the previous analysis (e.g., re-entering the data). Besides being flexible, versatile, and numerically dependable, a good interactive system should provide its users with a large variety of convenience features that are not feasible under a batch computing enviroment. For example, there should be internal documentations and help files so that a user need not have a User's Manual by his side to be able to make efficient use of the system; detectable spelling and logical errors made by the user should be detected by the system and facilities should be provided for the on-line correction of such errors; and there should be other error diagnostic and recovery features.

In this article, the design of an interactive system will be considered as the making of a sequence of decisions about the characteristics of the design, where early decisions may impose constraints on later decisions.

Whether a decision is made early or late in this sequence is not related to the importance of the system characteristic which the decision may affect. Therefore, if an early decision should place too much constraint on some criteria that are judged to be important, a designer may need to go through several iterations of the decision sequence as presented here.

Since I am much more familiar with existing interactive systems for statistical data analysis, I shall make references to them for illustration purposes. The concepts and considerations discussed in this article about the design of interactive systems are general in nature and are neither problem-specific nor dicipline-specific.

2. GENERAL CHARACTERISTICS UNDER CONSIDERATION

2.1 Portability

A portable system is one which can be run on a different computer or operating system (other than the one on which the system was designed and implemented) with little or no modification. A legitimate reason for considering a non-portable system is that if the system is designed to be run exclusively under a computer network enviroment, then by sacrificing portability, the designer may freely use non-standard features of the source language, operating system, or the host computer, to make optimal use of the available features. However, most of the existing interactive systems are severely limited in portability for the wrong reasons, the most common of which is that the designers did not take sufficient precaution at the design stage to make the system portable.

Recent literature on the evaluation of statistical software (see e.g., Francis, Heiberger, and Velleman [8], Plattsmier [14], and Velleman and Welsch [18]) generally considered Portability to be an important and desirable feature of any software. Therefore, a high priority should be placed on the criterion of portability although it will necessarily impose many constraints on other aspects of the system design. Portability of a system from maxicomputers to minicomputers poses many severe constraints on the design of an interactive system (see Ling [12]) and should probably be considered as an unrealistic goal to strive for at the present time.

2.2 Choice of Source Language

For the source code of an interactive system, an interpretive language, such as BASIC and APL, has many technical advantages over a language that requires pre-compilation, such as FORTRAN and PL/I. At the present time, portability considerations will limit the choice not only to BASIC and FORTRAN, but to some simple dialects of these languages, such as ANSI FORTRAN or a subset of the features in BASIC (see e.g., Isaacs [10]).

2.3 Choice of Data Structure and Program Structure

Once a programming language is decided upon, the designer should then choose a data structure and the corresponding program structure, taking into consideration the available core for the system and whether the system is to be general purpose or special purpose.

Decisions on the data structure include choosing the types of data the system will admit (scalers, vectors, matrices, arrays; binary-valued, integer-valued, real-valued, complex-valued, and categorical or non-numeric variables) as well as how they are stored, retrieved, and interfaced (fix-sized storage lacations for each type of data, variable-sized locations, or user-defined data structure with storage locations dynamically allocated). For example, for statistical data analysis, a commonly employed scheme is to allocate the bulk of the high- speed memory to a primary rectangular array of data values, classified as cases by variables. This array, together with other variables, arrays, and system parameters are stored in COMMON areas for passing data and system information among its subprograms. Such is the basic structure of interactive systems IDA [11], MIDAS [7], miniBMD [4], SIPS [9], and others. The SPEAKEASY system [17] (not designed to be portable) has perhaps the most general and flexible data structure of all interactive systems to date. When a user defines a variable under this system, he defines the type (real, complex, vector, matrix, etc.) as well as the size, and the system dynamically allocates storage locations for the variable, so that users using small datasets require less space than users using large datasets, which is generally not the case in other systems.

The most important considerations concerning the program structure of an interactive system are the degree of modularity of the system, the overlay structure of subprograms and utility programs (in FORTRAN) or the CHAINING or program-communication structure (in BASIC), and the isolation of machine-dependent codes (if they cannot be avoided) to a module of the system. Finally, the system should be designed and implemented in a way which makes

allowances for system expansion as well as extensions by its users who may wish to add modules of their own to the system.

3. USER INTERFACE

The aspects of user interface discussed in this Section are the chief distinguishing features of interactive systems from non-interactive ones. They pose the most challenging problems for the system designer and in my opinion, existing systems handle such problems with very limited degrees of success.

3.1 Control Language

A control language designed specifically for the novice in computing or a novice in the areas of application is likely to be too clumsy for the expert users. Conversely, a control language suitable for experts is likely to be too difficult for novices or new users. It is technically feasible to implement a flexible control language suitable to users of both extremes, although no existing system seems to meet the challenge in a completely satisfactory manner.

Consider some examples of how regression runs are specified in various existing statistical packages.

L1:	Regress (or Regr; System prompts for additional information - long prompts for users working in beginner's mode and short prompts for expert's mode)	IDA [11,13]
L2:	Regress var=5;1-3	MIDAS [6,7]
L3:	Regress,5,1-3	SIPS [1,9]
L4:	Regress y on (x1,x2,x3)	DATATRAN [3]
L5:	Multiregression(x1,x2,x3,y:r)	SPEAKEASY [17]
L6:	Regress y in c5 using 3 pred. in c1,c2,c3	MINITAB [15]
L7:	Regress c5 on 3 variables c1 c2 c3	MINITAB II [16]
L8:	Regress y on (principal-component (log(x1),log(x2),log(x3)))	DATATRAN [2]
L9:	eval cpm:=crossp(dtmx) eval cfs: rgqs(cpm) print coefs (where dtmx is the data matrix)	CS [5]

L1 has the advantage that the user has to remember only one control word (command word or keyword) for each procedure, and the system prompts for the rest. It's principal disadvantage is that it takes several commands and many prompt exchanges to accomplish the same task as that specified in a single line in L8.

L2 and L3 are efficient. They allow a user to specify a task with a minimal amount of typing. So are L4 and L5, to a lesser degree. However, all of them require the user to know the exact syntax, grammar, and where the commas, colons, simicolons and other symbols go. What if a user makes a typing mistake or syntatic error? The same question applies to L6 through L9 as well. None of the systems (L2-L9) has a satisfactory error recovery scheme.

In my opinion, an ideal control language should have L1 as the basic (or default) structure, but the user may override the prompting structure by specifying a task in some form similar to L2-L9, and if the user makes a syntatic error (implying he is not as familiar with the control language as he thought he was), the system reverts to the prompting mode automatically (which he may again override if he so chooses). This form of structure seems to be the most sensible way to design a control language. It does not make the unreasonable assumptions that a user familiar with some procedures of a system is equally familiar with other procedures; that a novice remains a novice; or that an expert does not make typing and syntatic errors.

L6 and L7 resemble a "natural" language (or ordinary English). They are definitely preferable to languages using cryptic abbreviations and unnatural syntaxes such as L9. However, the subtle trap is that what appears perfectly "natural" (once you are familiar with the keywords and the order of the parameter values) may have many equally "natural" equivalent expressions that the control language processor does not recognize. A case in point is the difference between the two natural languages L6 and L7 in MINITAB and MINITAB II respectively. Both are natural expressions for the same task but they are not compatible.

Of the languages illustrated here, DATATRAN admits the most general (and natural) expressions. The Consistent System (CS) is also very flexible in its control language L9. Its cryptic grammar and vocabulary are its chief drawbacks.

3.2 Internal Documentation and Help Files

Ideally, all documentation about an interactive system should be accessible by the user from the terminal while he is operating within the system. From the system

designer's point of view, such an implementation would be tedious to accomplish but it should present no technical difficulties. Frequently accessed documentation can be placed in core while lengthy or infrequently accessed documentation can be stored on disc or other secondary storage devices. Alternatively, documentations may be segmented as overlayed subprograms which would not occupy any active partition space until they are called. A User's Manual should be an optional or auxiliary feature of an interactive system, rather than a necessity.

In practice, the system that comes closest to this ideal is the SPEAKEASY system [17] which contains the equivalent of several hundred pages of printed documentation, hierarchically organized and retrievable through the use of the keyword HELP. IDA [11] has a very different internal documentation structure. Some documentations about the system are obtainable by executing commands COMM (for a list of the valid command words) and INFO (for general information about the system and various categories of commands). Other explanations can be obtained by the user as options to answering specific prompts. Most of the other existing interactive systems have very limited amounts of internal documentation.

3.3 Error Detection and Recovery

The simplest form of error detection is the insertion of source codes to check the user input for spelling and syntatic errors, and to produce appropriate diagnostic messages and provide on-line correction facilities for their remedy. A reasonable strategy is to assume that some user will make an error at some time at each place an error could occur, and have the system make checks for all possible errors. The implementation of this strategy entails a fair amount of effort of the programmer and a small additional amount of execution time, but saves the user a great deal of worrying about abnormal exits (or getting "bombed" out of the system).

Error detection need not, and probably should not, be limited to spelling and syntatic errors. Users often make logical errors that are easily detectable. For example, they may specify a logorithmic transformation on a variable which contains negative values; specify the same variable as two different independent variables in a regression problem (which would have led to the inversion of a singular matrix, if undetected); or many other syntatically correct specifications that would have led to execution errors that may be fatal to the interactive session. These errors should be detected before execution. On a more sophisticated level of error detection, an interactive system could be designed to detect "probable" errors of

application that are neither syntatic nor execution errors. For example, a user may fit a linear functional model to a set of data that are nonlinearly related; or there may be extreme outliers in the fitted model; or a user may ask to store some results in locations that would erase some of his data; and so on. In each of such instances, the condition of probable error can be detected and the user can be WARNED at the time of detection, so that he may either continue with the task as specified or change his task specification. The system IDA has a very elaborate subsystem of error detection codes that will detect all of the above types errors. As a result, a large portion of the source codes in IDA are error-detection codes. There is almost no limit to the amount of error-detection codes that can be usefully incorporated into an interactive system. The major practical constraints are the limited amount of human resources in writing such codes (as opposed to writing codes that carry higher priorities in developing a system) and possibly space limitations.

Error-detection codes are themselves of little value to the user unless they are accompanied by informative diagnostic messages and codes for easy recovery or change of tasks. Therefore, such codes are implied as necessary co-requisites of error-detection codes.

A different form of error recovery pertains to abnormal exits from the interactive system into the Operating System of the machine. This could occur when the user encountered execution errors (caused by machine arithmetic overflows or underflows) as the result of an undetected error in task-specification; or it could occur as the result of an inadvertent interrupt of the system by the user. In either case, there should be ways of recapturing the interactive session without having to reinitiate the entire session. Such a form of recovery is typically easier to accomplish when the system is coded in BASIC or some other interpretive language than in FORTRAN. In the former case, execution could easily be resumed at a specific line number of the source code near which the abnormal interrupt took place. In a FORTRAN enviroment, there is generally no easy way to get back into the system once execution is terminated, normally or abnormally. The most efficient way of handling error recovery of this type is to have special codes to intercept the pending interrupt (via machine language or assembler language routines) before it reaches the Operating System. However, such routines will make the interactive system machine-dependent as well as Operating-System-dependent and hence nonportable. For portable systems, a partial recovery is possible by periodically dumping the status of the system onto some temporary file, which can be used to bring the system up to the point of the dump, should there be a machine crash or some fatal execution error. IDA has such capabilities in the commands

HOLD and PICK. These commands can also be used to save the computation status at the end of one session, to be resumed from that point in a subsequent session, possibly at a later date, without having to retrace any previous computations.

4. IMPLEMENTATION CONSIDERATIONS

The items considered in this Section are design decisions relating to certain details in the implementation of software systems. Many of these decisions are not limited to the design of interactive systems, though some are.

4.1 Versatile Output Format

Batch systems generally have fixed output format for computational results because the standard printer page has 66 lines per page and can accommodate 132 printed characters per line (plus one carriage-control character). Interactive systems, on the other hand, are run on a large variety of terminals with different line widths and page lengths, ranging from the standard printer page to the short-width and short-length CRT screen. Attractively formatted output for one page-size is either unattractive or unsuitable for a different page-size, especially for plots. Consequently, it is highly desirable to write the printing and plotting routines of an interactive system with a varible page-size parameter which can be specified by the user, and have the system provide different formats for the same output depending on the value of the page-size parameter. It is also desirable to have the option for a user to specify the number of significant digits or decimal places in the printed output.

4.2 Versatile Plotting Options

Large size or full-page plots (even on small-sized pages) can take a considerable amount of time on a slow 10 cps teletypewriter or terminal. Often a much smaller-sized plot will contain sufficient detail for a particular problem. Other times, a user may wish to scan a moderate number of different small plots and use the large-sized plots of the same for only a few cases that require greater resolutions or details. IDA, for example, has three different sizes for scatter plots (large, small, and mini). Each of these plots requires approximately 80 sec., 30 sec., and 10 sec., respectively, to print on a 30 cps terminal. The advantage of having the option for a mini plot on a low-speed terminal seems obvious.

Other plotting versatility considerations include a flexible choice of scales and labels for each plot, and the incorporation of interface facilities for selected plots on various standard plotting equipments such as the CALCOMP pen plotter and the Tektronics CRT graphic unit.

4.3 Optional Background Output

Since an increasing number of interactive system users are computing on CRT terminals (which can be operated in much higher output speeds and at a lower cost than hardcopy terminals), provisions should be made for users to selectively save the output of part or all of the terminal session onto a data file in order to obtain copies of these results on a hardcopy printing device (such as a standard high-speed printer) at the end of a session. Facilities for such a mode of background output can be useful even for the user whose foreground output is on a hardcopy terminal, when multiple copies of selected portions of the results are desired.

4.4 Other Considerations

There are many other criteria and options that must be taken into consideration in the design of a computational system, e.g., the numerical accuracy and computational efficiency of the algorithms, the efficiency of programming codes; whether users have the option to choose the degree of numerical accuracy in the computations, whether the results are guaranteed to be accurate to the number of digits printed, and so on. However, since these considerations apply to interactive and non-interactive systems alike and they pertain to the finer details of such systems, they will not be elaborated here.

5. CONCLUDING REMARKS

Once an interactive system has been written, even with provisions for modifications and extensions, it is without exception very difficult to alter the basic structure and characteristics of the system. Therefore , if an interactive data analysis system is contemplated, the designer must pay close attention to all of the potential features in the system, whether some of those features are intended to be included in the initial implementation of the system or not. In this article, the design process is presented as a sequential decision process. Most of the features discussed are considered by this author to be

either necessary or highly desirable features in a good interactive system. They reflect the author's extraction and extrapolation of features in existing interactive systems.

The criteria implied by, and derivable from, the discussion of these features are consistent with the generally accepted criteria in the evaluation of interactive systems and therefore can be used as such. However, the purpose of the article is to give an account of the general considerations in the design of an interactive system and to focus attention on certain areas where there seem to be much room for improvements in the existing interactive data analysis systems.

6. REFERENCES

[1] Avery, K.R. and Avery, C.A., "Design and development of an interactive statistical system (SIPS)," Proceedings of Computer Science and Statistics: 8th Annual symposium on the Interface, Health Sciences Computing Facility, UCLA, 1975, 49-55.

[2] Brode, J., "Generalizing the function call to statistical routines-an application from the DATATRAN language," Proceedings of the Computer Science and Statistics: 10th Annual Symposium on the Interface, National Bureau of Standards, Gaithersberg, Maryland, 1977.

[3] Brode, J., Stamen, J., and Wallace, R., "The DATATRAN language," Proceedings of the American Statistical Association, Statistical Computing Section, 1976, 126-9.

[4] Buchness, R. and Engleman, L., "MiniBMD: A minicomputer statistical system," Proceedings of the Computer Science and Statistics: 10th Annual Symposium on the Interface, National Bureau of Standards, Gaithersberg, Maryland, 1977.

[5] Dawson, R. and Klensin, J., "Data analysis in the Consistent System," unpublished manuscript, M.I.T., 1976.

[6] Fox, D.J., "Some considerations in designing an interactive data analysis system," Proceedings of the Computer Science and Statistics: 8th Annual Symposium on the Interface, Health Sciences Computing Facility, UCLA, 1975, 61-5.

[7] Fox, D.J. and Guire, K.E., Documentation for MIDAS, revised 2nd edition, Statistical Research Laboratory, University of Michigan, August, 1974.

[8] Francis, I., Heiberger, R.M., and Velleman, P., "Criteria in the evaluation of statistical program packages," American Statistician 29 (February 1975), 52-5.

[9] Guthrie, D., Avery, C., and Avery, K., Statistical Interactive Programming System (SIPS), User's Reference Manual. Oregon State University Bookstore, Corvallis, Oregon, 1974.

[10] Isaacs, G.L., "Interdialect translatability of the BASIC programming language," ACT Technical Bulletin No. 11, The American College Testing Program, Iowa City, Iowa, 1972.

[11] Ling, R.F., "IDA (Interactive Data Analysis)," Department of Mathematical Sciences Technical Report No. 237, Clemson University, 1976.

[12] Ling, R.F., "Constraints in the design and implementation of interactive statistical systems for minicomputers," Proceedings of the Computer Science and Statistics: 10th Annual Symposium on the Interface, National Bureau of Standards, Gaithersberg, Maryland, 1977.

[13] Ling, R.F. and Roberts, H.V., "IDA and user interface," Proceedings of the Computer Science and Statistics: 8th Annual Symposium on the Interface, Health Sciences Computing Facility, UCLA, 1975, 91-4.

[14] Plattsmier, R.A., "Criteria for evaluation of interactive statistical programs and packages," Proceedings of the Computer Science and Statistics: 10th Annual Symposium on the Interface, National Bureau of Standards, Gaithersberg, Maryland, 1977.

[15] Ryan, T.A., Joiner, B.L., and Ryan, B.F., MINITAB Handbook. North Scituate: Duxbury Press, 1976.

[16] Ryan, T.A., Joiner, B.L., and Ryan, B.F., "Minitab II, 1977," unpublished manuscript, 1977.

[17] SPEAKEASY-3 Reference Manual Level Lambda IBM OS/VS Version, compiled by Cohen, S. and Pieper, S.C., Argonne National Laboratories, Argonne, Illinois, 1976.

[18] Velleman, P. and Welsch, R.E., "Some evaluation criteria for interactive statistical program packages," Proceedings of the American Statistical Association, Statistical Computing Section, 1975, 10-2.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: N 93	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): General Considerations on the Design of an Interactive System for Data Analysis		5. TYPE OF REPORT & PERIOD COVERED: Technical Report
		6. PERFORMING ORG. REPORT NUMBER: TR 269
7. AUTHOR(s): Robert F. Ling		8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0451
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Clemson University, Dept. of Mathematical Sciences, Clemson, South Carolina 29631		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 042-271
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Code 436, Arlington, Va. 22217		12. REPORT DATE: NOV. 1977
		13. NUMBER OF PAGES: 12
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Invited paper presented at the TIMS/ORSA Annual Meeting in Atlanta, Georgia, November 8, 1977.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Design of computer software; Interactive system for data analysis.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Among the most important criteria in the design and implementation of an interactive system for data analysis are: data structure, control language, user interface, system versatility, extensibility, and portability. The design of an interactive system, viewed as a sequential consideration of these criteria, is discussed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)