[Usenix, July 1982]

The UCSD MSG System: Iterative Design in the UNIX Environment

Philip J. Mercurio

Cognitive Science Laboratory

University of California, San Diego C-015

La Jolla, Ca 92093

UUCP: ucbvax!sdcsvax!mercurio

ARPA: sdcsvax!mercurio@nprdc

Abstract

This paper is an investigation into the process of design of a user interface. The starting point was a fully implemented electronic mail software package called the "MSG System". The design technique employed is an iterative one—feedback from a community of test users shaped each successive version of the software. This iterative design technique has produced a user interface preferred by many users of the original version. The design process is discussed, along with some of the more important features that were developed. The technique with which the introductory user manual was structured is described. Limitations of the iterative design technique and implications for a principled design are also discussed.

The research reported here was conducted under Contract N00014-79-C-0323, NR 667-437 with the Personnel and Training Research Programs of the Office of Naval Research, and was sponsored by the Office of Naval Research and the Air Force Office of Scientific Research.

Requests for reprints should be sent to Philip J. Mercurio at the address given above.

The MSG Project

There are two phases to the construction of any system, design and implementation. It is rare that the designer has a fully implemented system to experiment with. The goal of this project is an investigation of the process of user interface design, given a fully functional software system as a starting point. The design technique employed is an iterative one— feedback from users of the system has been used to evaluate each successive version of the software.

The UCSD MSG System consists primarily of two programs for the reading and sending of electronic mail, "msg" and "snd". These programs were originally written at the Rand Corporation, and have been re-worked by several people at UCSD. The version I began working on a year ago, what I'll be referring to as the "original" version, was completed in 1980 by Greg Haerr and has been in widespread use on several computers at UCSD since then.

The MSG System programs have a number of user interface features which have caused them to be preferred over the other UNIX[1] electronic mail systems[2] available at UCSD. Their most important advantage is that they are "hot key" systems—the pro-grams provide immediate feedback after each keypress, whenever possible. All commands in "msg" and "snd" are entered by typing their one-character abbreviations. The programs respond immediately by completing the full name of the command. This feature has two advantages: 1) it makes the command abbreviations easier to learn, and 2) it assures the user that the system is still active.

The original MSG System also has documentation written for the novice computer user. However, new and occasional users encounter many difficulties in learning the user interface given the available documentation. During the course of this project not only has the software been re-worked, but the introductory documentation also has been re-written.

Structure of the User Interface Design

The design of the user interface has proceeded in two phases. In the first phase, suggestions and complaints regarding the original version were casually solicited from daily users of the system. Many valuable suggestions arose from discussions among members of our laboratory who are expert users of the sys-tem. These ideas were then assembled into the first new version. Although this phase produced a more powerful software tool, it did not produce one that was necessarily easier to use.

The second phase of the project centered around a group of about a dozen test users of the system, again members of the laboratory. These people were all experienced users of the original MSG who volunteered to use the new, experimental version of the program for at least some of their daily electronic mail handling. Most of these users switched to using the new system exclusively. They subjected each new version of the user interface to extensive testing. Electronic mail has been used as the primary means of communication within the test user community, facilitating convenient and voluminous discussion of all aspects of the system.

New Features

User Interface

A number of valuable features came out of this second phase:

Feedback on Irrevocable Actions. The original version gave only partial feedback regarding irrevocable actions, such as discarding messages, which require confirmation from the user. The new version attempts to be clearer in describing such actions. Warning and error messages are worded with the assumption that the user is familiar with a few basic concepts regarding electronic mail which are discussed in the introductory user's manual.

Customized Interfaces. As the size and variety of the test user community grow, so do the requests for new features. One means of dealing with this problem is via customized user inter-faces. Using a string variable handling package written by Gary Perlman at UCSD, users are able to specify options which tailor the functioning of the MSG System programs. Whenever a new feature was called for by a significant portion of the test user community, it would be installed as an option. In all cases, the default value for an option is the one which provides more information to the user or forces the user to make a choice. Novice users are thus presented with the system in its most verbose state.

---

[1] UNIX is a trademark of Bell Laboratories.

[2] The MSG System is fully compatible with the standard UNIX mail programs "mail" and "delivermail".

The problem with this approach is that it tends toward anarchy. Without initial specifications regarding the scope of the system, it becomes difficult to inhibit the proliferation of new options.

On-Line Help Messages. The amount of instructional information available to the user has been greatly increased. On-line help now exists at five levels:

1) Menu of Commands. A menu mapping single characters to their expanded command names is presented to the user when either of the MSG System programs is entered. The menu is also displayed when the user types a '?' for help.
2) One-Line Command Descriptions. A list of brief, one-line descriptions of each command's function is available by typing a '?' followed by a RETURN. This list was the only form of on-line documentation available in the original version.
3) Detailed Command Descriptions. The user may obtain a detailed description of any command by typing a '?' followed by the single character abbreviation for the command. Each of these command descriptions contains information regarding the primary function of the command, it's usage and side-effects, and pointers to related commands. This, in conjunction with the menu of commands, provides the user with the opportunity to expand his or her knowledge of system's various functions.
4) Context-Dependent Help. The usage of the above three forms of documentation has been generalized so that the user may request help at any time that the program is awaiting input by typing a '?'. If the user is being asked to confirm an irreversible command, typing a '?' provides more information on the consequences of that action. When the program is expecting arguments for a command, a '?' obtains help on valid arguments.
5) Complete Users Manual. The user may gain access to the complete user's manual by typing two question marks ('??'). This complete document, which will contain tutorial introductions to each of the MSG system programs as well as instructions for advanced users, is in the process of being written. The user will be presented with a table of the contents of this document and allowed to select those portions in which she or he is interested. I have been working with Robert Brien of the Department of Technological Instruction, Universite Laval, in Quebec, on structuring the introductory manual. I will discuss this process in more detail below.

Additional Enhancements.

A number of additional enhancements to the internals of the MSG System have been incorporated:

Limited ARPANET Compatibility. Features were added to increase compatibility with the ARPANET's RFC 733 standards for electronic mail, such as the addition of "date" and "reply-to" lines in the headers of outgoing messages. Attempts at complete compatibility with the ARPANET standards were felt to be beyond the scope of this project.

MSGNET Aliases. Users on any of a number of different computers running the MSG System all have access to a common database of aliases for individuals and groups. This database is referred to as a MSGNET. I, for example, will receive a message addressed to "mercurio" sent from any of five UCSD computers in our local network. This database is maintained by a couple of key people referred to as the "postoffice". Users may send messages to the address "postoffice" in order to establish an alias or to be added or deleted from distribution lists (called "memo groups"). The database is stored in a simple, easily modified format and converted into an alias file for the "delivermail" program[3] by a program called "makealias". This allows our aliases and memo groups to be addressed by users on remote ends of the network regardless of which mail-handling program they employ. For example, the entire Cognitive Science Lab may be addressed on the UUCP network via "...!ucbvax!sdcsvax!csl". Only one copy will be sent across the network until it reaches "sdcsvax", where it will be distributed to the entire group. The message could be redistributed from any one of the computers in the UCSD MSGNET. Software is being developed to make the maintenance of a MSGNET database more automatic and centralized.

Machine Aliases. The development of the MSGNET software required establishment of a database of network paths to various local machines. This database has been expanded and made accessible to the user. By specifying an address as:

[3] This is the UNIX system program which performs the task of delivering a properly formatted message to the recipients, both locally and across various networks.

```
<machine name>?<user name>
```

the full network route to "<machine name>" would be substituted

by the "snd" program. For example, I can address a friend at Columbia University with

```
cubs45?noel
```

and the "snd" program would translate that into

```
ucbvax!chico!learned!cubs45!noel
```

This feature will probably be discarded in favor of the enhance-ments to "delivermail" being developed independently by Steve Bellovin at the University of North Carolina and Tim Curry at UCF.

Preparation of the Introductory User's Manual

The introductory user's manual for the MSG System is being created using techniques for the design of instructional text developed by Robert Brien during his recent visit to the Cognitive Science Lab.[4] We began with an analysis of the subject material. First, a list of the tasks that an educated user should be capable of performing was created. This list was later used to define a set of instructional objectives used to evaluate the course materials. Each task was then represented in the form of a procedural schema.[5] Each schema consists of a state in an expert user's interaction with the system (called the condition), the user's goal at this point, the action that the expert user performs to achieve this goal, and the result of this action. For example, here is a procedural schema for printing a message:

Condition	**Goal**	**Action**	**Result**
msg command level	read third message	press 't' ("type") '3' and RETURN	message 3 printed

Given such a list of the schemas that need to be installed in the student's mind, the next step is to perform a backwards analysis on the set of schemas. Each schema is broken down into its component concepts. The basic concepts which underlie these are then assembled into a superordinate schema. This process is iterated until schemas believed to be already possessed by the naive user are reached. Here is a schema superordinate to the one given above:

Condition	**Goal**	**Action**	**Result**
msg command level	read a particular message	give "type" command and argument	message printed

With these superordinate schemas in mind, the next step is to look for analogies which employ these schemas. These analogies are then assembled into a primitive mental model of how the system functions. Instructional text that will convey this model is then written. This method not only provides a systematic way of developing the ideas covered in an instructional text, but it also has implications for the sequence in which these ideas should be presented.

In the actual writing of the manual, Brien, who is not an expert user of the MSG System, would write the first draft, and I would fill in as the subject-matter specialist. We chose to present each command in the form of a schema, first presenting a condition in which the command would be used, the goal which the command achieves, the actions taken to execute the command, and the resulting state of the system. The introductory manual has been divided into 5 sections called "modules". The first module addresses the basics of "What is a computer?", while the second serves as an introduction to electronic mail in general. The last three modules are intended to be worked through in front of a terminal. Module 3 gives instruction on logging onto the computer, and modules 4 and 5 are introductions to using the pro-grams "msg" and "snd", respectively. Subsequent modules will be written to deal with topics for advanced users.

Once the first draft was complete we began formative evaluation. Undergraduate subjects were given the course materials and access to the computer. They were then tested using the course objectives developed earlier. These test results and the subjects' comments are being employed in the development of subsequent drafts of the course materials.

---

[4] These techniques are discussed in two papers currently in preparation: "Mental Models: Suggestions for Their Design", Brien and Mercurio (1982) and "Sequencing Instruction: A Cognitive Science Perspective", Brien (1982).

[5] See Rumelhart (1979) for a discussion of schemas.

Although initial results appear very positive, this process will need to be iterated several times before the course is complete.

Lessons Learned from The MSG Project

In this project, the primary criteria employed in user interface design decisions has been feedback from users of the system. Since the project began with a complete system which had been in use for several years, there are no underlying principles organizing the design. No amount of iterative re-designing can produce a principled design—design principles cannot be grafted onto an existing system. There are four design principles which should have been incorporated into the MSG System:[6]

Conceptual Model. The foremost aspect missing from this project is a consistent conceptual model underlying the design of the system and the image it should present to the user. An example of a model which might be used with an electronic mail package is that of a secretary who sorts and summarizes mail you've received, and formats and addresses your outgoing mail. Given such a model, a system might be designed which performed the same sorts of tasks that a secretary might.

Task Domain. A specification of the domain of tasks the system should be capable of performing is necessary in order to define the scope of the design. Without such a specification, it is difficult to decide which of the features requested by users should be incorporated into the design.

System Image. Many ideas arose during the development of the introductory manual that were incorporated as changes in the programs themselves. While many of these changes were minor, a few pointed out major inconsistencies in the image of the system as presented by the user inter-face. Ideally, a consistent system image should be developed which is conveyed not only in the documentation but also in every aspect of the user's interaction with the system. This system image should faithfully represent the underlying conceptual model upon which the design is based.

Plan for Total Revision. The MSG System has provided an excellent medium in which to develop a new user inter-face. The original version took the place of a mock-up of the system, with the added advantage that it was a fully functional system before the design process even began. However, it now needs to follow the path of all good first versions—it needs to be discarded. While many sound ideas have been developed in the course of this project, so have many inconsistencies. No amount of post hoc design can be sufficient to overcome the inertia of an existing design. A system as complex as MSG should be developed with the intention that, at some point, the existing software will be revised totally.

REFERENCES

Brien, R. & Mercurio, P. J. *Electronic Mail with the UCSD MSG System.* La Jolla, California: Center for Human Information Processing, University of California at San Diego. June, 1982. (Cognitive Science Laboratory user manual)

Dick, W. *Formative Evaluation*. In L. J. Briggs (Ed.), Instructional Design: Principles and Applications. Englewood Cliffs, N. J.: Educational Technology Publications, 1977.

Norman, D. A., *The Trouble with UNIX*. Datamation, vol. 27, #12. November 1981.

Norman, D. A., *Some observations on mental models*. In D. Gentner and A. Stevens (Eds.), Mental Models. Hillsdale, N. J.: Erlbaum Associates, 1982a.

Norman, D. A., *Steps toward a Cognitive Engineering: Design rules based on the analysis of human error*. Proceedings of the Conference on Human Factors in Computer Systems. Gaithersburg, Md., March 15-17, 1982b.

[6] See Norman (1982b & c) for more complete discussion of these principles.

Norman, D. A., *How to make computer systems that people will like to use: steps toward a Cognitive Engineering.* Paper presented at Western Electronic Show and Convention, Anaheim, California, September, 1982c (Wescon/82).

Perlman, G. P., *Two papers in Cognitive Engineering: The design of an interface to a programming system & Menunix: a menu-based interface to UNIX (user manual).* La Jolla, California: Center for Human Information Processing, University of California at San Diego. November, 1981. (Office of Naval Research technical report no. 8105)

Rumelhart, D. E. *Schemata: the building blocks of cognition*. In R. Spiro, B. Bruce, & W. Brewer (Eds.), Theoretical Issues in Reading Comprehension. Hillsdale, N. J.: Erlbaum Associates, 1979.

Stolovitch, H. D., & Berthelot, S. *Rapid verification and revision of the readability and comprehensibility levels of instructional text.* Paper presented at the twenty-first annual conference of the National Society for Performance and Instruction, San Diego, California, April 1982.