CAN WE EXPECT TO IMPROVE TEXT EDITING PERFORMANCE?

David W. Embley and George Nagy

Department of Computer Science
University of Nebraska
Lincoln, NE 68588

INTRODUCTION

The subject of human factors in computing systems has become an intriguing and exciting field of study. In the ten years since the publication of Weinberg's book, The Psychology of Computer Programming [17], the field has grown considerably as is evidenced by Shneiderman's Software Psychology [16], which surveys current work and points in directions research might continue. One anthology containing many representative projects is Coombs and Alty's Computing Skills and the User Interface [6].

One of the active areas is computer text editing. At the Xerox Palo Alto Research Center Card, Moran, and Newell have conducted several studies on the psychology of computer text editing [2, 3, 4, 5]. For her dissertation research at Stanford University and in cooperation with the research group at Xerox, Roberts has conducted several experiments to evaluate human factors in computer text editors and has suggested numerous others [15]. Ledgard and his colleagues at the University of Massachusetts have investigated natural language aspects of text editor command languages [14]. At the University of Illinois Hammer and Rouse have investigated freeform text editing behavior [12], and more recently Hammer has completed his dissertation on human aspects of text editing [13].

For some time now we have also been interested in studying human factors aspects of computer text editors. We have surveyed the literature [9], and we have conducted several investigations of our own [10] ranging from an application of file-comparison algorithms in editor research [1], through prediction of editing performance [7], to the design and implementation of SIMPLE, our own editing system for beginners [8, 11].

Currently we are concentrating our efforts on gathering data to determine how much time users spend performing various editing activities. We intend to extend the work begun in previous studies, to investigate suboptimal editor performance including errors and nonoptimal means of achieving goals, and to study the broader aspects of editing such as file manipulation and job control that have largely been ignored.

OBJECTIVES

A study of text editors and the editing process can lead in several directions. Among the many alternatives, we have chosen to work on the objectives listed below. These are ordered with short-term objectives first. Whether the latter objectives are achieved or even attempted depends on our success in achieving some of the earlier ones.

- Determine the principal components of task time distribution. Where do users spend their time? editing? manipulating files? seeking help? How does the time distribution change as a user gains expertise? performs different types of editing tasks? uses different editing systems?

- Study suboptimal performance. To what extent do errors increase task completion times? What constitutes an error? typos only? selection of a lengthy editing sequence when a shorter one would have accomplished the same task? undoing previous work? To what extent can training or on-line or off-line help reduce error?

- Predict editing task duration. For a given user, system, and task, how accurately can the task time be predicted? What are the essential characteristics that must be observed and measured?

- Validate models of editing performance proposed by others. How accurately and to what extent do these models predict performance?

- Develop realistic models of editing activities that incorporate aspects neglected in previous studies, such as file manipulation and error correction. If these more complex activities are also modeled, how

accurately can task time be predicted?

- Compare editor quality. For a given type of user and classification of tasks, which editor or version of an editor is best? What are quantitive measures of editor quality for given subjects, tasks, and systems? How easily can these quantitative measures be obtained?

- Improve guidelines for editor design. What kind of guidelines can be provided? To what extent is it necessary to know the type of user and classification of editing tasks in order to make appropriate design decisions?

- Provide data to facilitate editor standardization. What type of quantitative data would be useful for encouraging standardization? How should the data be utilized?

- Provide quantitative information to assist in training, learning, documentation, and help. What information is likely to be most helpful? Can sophisticated online help routines be implemented that not only provide the "right" help but also suggest to users how their performance might be improved? Can help features be improved by taking advantage of current research on "expert system"?

- Lay quantitative foundations for the study of cognitive processes involved in editing activities. What data would be interesting for the cognitive psychologist? How can the data be appropriately presented?

METHODOLOGY

Our experimental world consists of subjects who perform editing tasks using one of several editing systems. Both field experiments, which do not substantially affect the subjects' normal behavior, and controlled experiments, where certain constraints are imposed on the subjects' activities, are performed. The subjects' behavior is observed essentially through a "gremlin" hidden in the editing terminal who records and time stamps each editing transaction for subsequent computer analysis. The files that contain the subjects' work are also saved both before and after the session.

Subjects

The subjects in the field experiments are generally students in an introductory programming course (not taught by the experimenters) who are preparing their programming assignments. At the beginning of the course a logon message warns the entire class that some sessions may be recorded for experimental purposes, but there is no other indication of when experimental data collection is in progress. Students may choose not to be monitored if they wish. Subjects are selected for field experiments by random choice of account number from among willing participants who are actively involved in class programming assignments. The same subject may be observed several times.

In the controlled experiments the subjects are students of varying degrees of programming and editing expertise. They receive a flat fee for their participation.

Editing Tasks

An editing task is defined as the transformation from the state in which the subject's files are at the beginning of the observation period to the state they are in at the end of the period. In the case of initial program entry, the source file may be null. An entire session may be divided into several observation periods delimited, for instance, by a subject's attempts to compile and execute the program under preparation. By scheduling the observations throughout the course of the semester, editing tasks of varying difficulty may be obtained.

Eventually, natural language (document preparation) tasks performed either on an office word-processing system or on a general-purpose computer are also to be studied. We intend to select such tasks also by capturing part of the normal workload of the subjects.

Editing Systems

The editors available to us include the SIMPLE editor for novice programming students, IBM's CMS EDIT (imbedded in CP), and the AM Jacquard J100 multiterminal word processor. At this time only the SIMPLE data collection program is fully operational. The CMS instrumentation is ready, but requires additional testing and adjustment. Programs for the J100 are likely to be more difficult to implement.

In order to analyze the significant aspects of the person-machine interaction in each of these systems, we record commands (and responses) for access to files, common utility programs, and compilers and interpreters in addition to commands for symbol manipulation within a single file.

Data Collection and Analysis Programs

The experimental designs are built around the concept of automated analysis of the time-stamped transcripts of editing sessions performed under various conditions on one of several systems. The principal data collection and analysis programs consist of the following:

MONITOR This program captures each line entered or displayed on an editing terminal and saves it in a file. The time at which an entry is

completed and the time at which the computer response is completed are also recorded in the same file. The time is measured to one one-hundredth of a second. A subroutine in the MONITOR program, called STEAL, saves the work files before and after the session.

LEXICAL ANALYZER This program recognizes and labels legal editor commands in each line of entry, flags error conditions, counts the number of characters in each command and argument and the number of lines and characters in the response, and computes the command entry time and computer response time for each command. The input of this program is the file generated by MONITOR and its output is a file consisting of labelled and flagged commands with certain line and character counts, pre- and post-command time intervals, and response times.

SESSION ANALYZER This program compiles statistical information for a session. The statistics include the time intervals, frequencies, command character counts, argument character counts, and error conditions associated with each command or predetermined group of commands. In addition to session totals, means and standard deviations are computed. The input of this program is the file generated by the LEXICAL ANALYZER. The output is a file containing the statistics and a set of printed tables.

MULTI-SESSION ANALYZER This program, which is currently at the design stage, statistically compares several sessions consisting of the same subject performing different tasks, different subjects performing the same task, or the same subject performing the same task on different editing systems. The input of this program is the file generated by the SESSION ANALYZER, and the output is a file of suitable statistics.

Experimental Protocols

Once an experimental editing task has been obtained from a field observation of an actual work session, other subjects may execute the same task under various controlled conditions. Among the pilot experiments performed to date are the following:

- A subject is given listings of the source and target files where the necessary modifications are marked with proofreaders symbols and asked to make the required changes using the editor.
- The string of editing commands obtained in the field experiment, in the controlled experiment described above, or by a "committee of experts" is provided in the form of a listing and a subject is asked to make the changes by entering the commands as quickly as possible using the editor.
- In order to eliminate the effect of the computer responses, a subject is asked to enter the string of editing commands obtained in one of the above ways using only the INSERT (or ENTER) mode of the editor.

Innumerable variations are possible. We may ask subjects to optimize measures other than time such as the number of commands or the number of keystrokes. We can also restrict the editing commands available in each experiment. Subjects may also be selected according to several criteria. It is clear that the design of experimental protocols that lead to the greatest insights using the available instrumentation will continue to present a challenge to our ingenuity. It is expected that the current exploritory experiments will generate a number of specific hypotheses that can be tested using new data.

INITIAL OBSERVATIONS AND DISCUSSION

After conducting a few initial pilot experiments, it became obvious that the overall experiment is quite intricate and complex and that the amount of data collected could become massive. There are several recondite programs involved. Each of these must be debugged and thoroughly tested. This is especially difficult for programs that have hooks into the operating systems to gather time stamps because the operating systems are being continuously changed and updated. Numerous special cases and quirks of the systems have arisen; for example, internal hex codes for time stamps have been "swallowed" as control characters, special commands such as "null input" have short-circuited the system and avoided the time-stamping code, and some special system modes cannot be time stamped at all.

Now that many of the routines have been written, it has become easy to generate numerous output products, both printouts and files. Since keeping track of experimental data is already becoming burdensome, an elaborate, automated naming convention has been established to identify each product in a meaningful way. Deciding what to discard and what to keep is also difficult. Since we are not yet sure which data is of interest, we feel uncomfortable destroying any raw data. Eventually we will have to summarize the raw data into some more compact form.

By acting as subjects ourselves in an attempt to generate an "optimal" editing sequence for a given source and target file, we learned that we don't even know how to characterize "optimal". We attempted to minimize the number of keystrokes, and although successful, we are convinced that the editing sequence produced was not "optimal" because we had to spend a great deal of time figuring out whether one sequence would have fewer keystrokes than another. For the particular example, an unusual combination of editing commands was required to minimize the number of keystrokes. We suspect that unusual combinations may be more of a rule than an exception if the

definition of "optimal" is in terms of keystroke minimization.

Actual data that has emerged from the experimentation to date is extremely tentative and is based on only one or two pilot runs. We have observed that in an uncontrolled experiment the average entry time (between the end of the previous computer response and the completion of the command) is 20 seconds. The average time for the computer response in the same experiment is 2.5 seconds. The command with the longest average command entry time (27 seconds) is FORWARD; that with the shortest (3 seconds) is TYPE.

A subject took 53 minutes to complete a particular, actual editing task. A second subject made the same changes using a copy marked up with proofreaders symbols in 9 minutes. Just keying in the command sequence used by the original subject took 7 minutes.

The original subject used 94 legal commands (810 keystrokes) and 3 illegal commands (89 keystrokes). The subject working off the marked up copy used 53 commands (457 keystrokes). The optimal sequence established by a committee of "experts" instructed to minimize keystrokes required just 30 commands and 266 keystrokes. It should be noted that each of the "experts" took about an hour to devise a command sequence, so it is questionable to what extent the command sequence may be considered "optimal".

EXPECTATIONS

Since the first step of our research program is to determine where the bulk of the time goes in editing, we are unable to set a firm direction until this is accomplished. We do not, however, expect to find major differences among editors or opportunities for significant improvement in editor design for routine tasks performed optimally. Instead, we believe that we will have to concentrate on suboptimal performance and on the prevention of costly errors through improved editor design, on-line and off-line documentation, and training.

Our experiments should, however, lead to improved choice of parameters and possible validation of existing models of editing performance such as the GOMS and keystroke models [4, 5]. This, in turn, will lead to improved time estimates for editing tasks under a broad range of conditions.

We hope that our experimental results, and possibly our data collection programs, can eventually be made available in a form useful to other research groups. A step in this direction is an embryonic cooperative research project with a group at the University of Liverpool, where we intend to combine our quantitative observations with qualitiative observations of the nature of the subjects' understanding of editor structure in order to develop expert "help" systems.

ACKNOWLEDGMENTS

We are grateful to former and present students Tom Lee, Chingwha Chang, Barry Rajaidehkordy, Iraj Mohabelian, Al Shanmugam, and Kevin Kelle for carrying out the programming for data collection and analysis and for calling our attention to numerous oversights. Steve Wengel, a recent psychology graduate, is in charge of our controlled experiments. Jerry Weinberg has been an irregular participant at our weekly project meetings and has offered both moral support and many helpful suggestions. We also acknowledge a long discussion with Ben Shneiderman which helped to focus some of our objectives, and several lively exchanges with Michael Coombs of the University of Liverpool who spent Summer 1981 with us. Finally, we thank the University of Nebraska Research Council for financial support.

REFERENCES

1. Anandan, P., Embley, D.W., and Nagy, G. An application of file-comparison algorithms to the study of program editors, International Journal of Man-Machine Studies, Vol. 13, No. 2, August 1980, 201-211.

2. Card, S.K., Moran, T.P., and Newell, A. The manuscript editing task: a routine cognitive skill, Xerox Research Rep. SSL-76-8, Palo Alto Research Center, Palo Alto, California, December 1976.

3. Card, S.K. Studies in the psychology of computer text editing systems, Ph.D. Thesis, Department of Psychology, Carnegie-Mellon University, Pittsburgh, Pennsylvania; also Xerox Research Rep. SSL-78-1, Palo Alto Research Center, Palo Alto, California, August 1978.

4. Card, S.K., Moran, T.P., and Newell, A. Computer text editing: an information-processing analysis of a routine cognitive skill, Cognitive Psychology, Vol. 12, 1980a, 32-74.

5. Card, S.K., Moran, T.P., and Newell, A. The keystroke-level model for user performance time with interactive systems, Communications of the ACM, Vol. 23, No. 7, July 1980b, 396-410.

6. Coombs, M.J. and Alty, J.L. (Eds) Computing Skills and the User Interface. Academic Press, London, 1981.

7. Embley, D.W., Lan, M.T., Leinbaugh, D.W., and Nagy, G. A procedure for predicting program editor performance from the user's point of view, International Journal of Man-Machine Studies, Vol. 10, No. 6, November 1978, 639-650.

8. Embley, D.W. and Nagy, G. SIMPLE specifications, Department of Computer Science Rep., University of Nebraska, Lincoln, Nebraska, June 1979.

9. Embley, D.W. and Nagy, G. Behavioral aspects of text editors, Computing Surveys, Vol. 13, No. 1, March 1981a, 33-70.

10. Embley, D.W. and Nagy, G. Empirical and formal methods for the study of computer editors, in Computing Skills and the User Interface, M.J. Coombs and J.L. Alty (eds.), Academic Press, London, 1981b, 465-496.

11. Embley, D.W. and Nagy, G. SIMPLE - a programming environment for beginners, SIGCSE Bulletin, Vol. 13, No. 4, December 1981, 7-12.

12. Hammer, J.M. and Rouse, W.B. Analysis and modeling of freeform text editing behavior, Proceedings of the 1979 International Conference on Cybernetics and Society, Denver, Colorado, October 1979.

13. Hammer, J.M. The human as a constrained optimal text editor, Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana Illinois; also Coordinated Science Laboratory Rep. T-105, University of Illinois, Urbana, Illinois, June 1981.

14. Ledgard, H., Whiteside, J.A., Singer, A., and Seymour, W. The natural language of interactive systems, Communications of the ACM, Vol. 23, No. 10, October 1980, 556-563.

15. Roberts, T.L. Evaluation of computer text editors, Ph.D. Thesis, Department of Computer Science, Stanford University, Stanford, California; also Xerox Research Rep. SSL-79-9, Palo Alto, California, November 1979.

16. Shneiderman, B. Software Psychology, Winthrop, Cambridge, Massachusetts, 1980.

17. Weinberg, G.M. The Psychology of Computer Programming, Van Nostrand Reinholt, New York, 1971.