AD-A104 827 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2
EXPERIMENTAL EVALUATION OF THE ZOG FRAME EDITOR.(U)
APR 81 C K ROBERTSON, D L MCCRACKEN, A NEWELL N00014-76-C-0874
UNCLASSIFIED CMU-CS-81-112 NL

END
DATE FILMED
10-81
DTIC

CMU-CS-81-112

Experimental Evaluation of the ZOG Frame Editor

C. Kamila Robertson
Donald L. McCracken
Allen Newell

21 April 1981

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-81-112

Experimental Evaluation of the ZOG Frame Editor

C. Kamila Robertson
Donald L. McCracken
Allen Newell

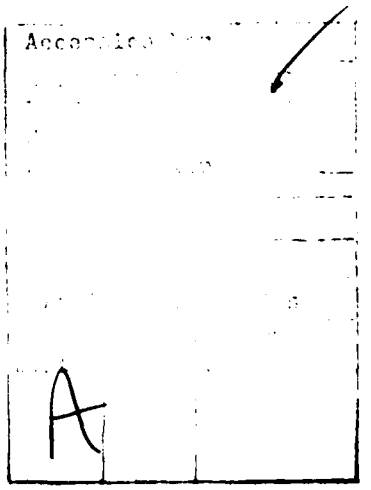
21 April 1981

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

This paper will appear in the
proceedings of the annual conference of the
Canadian Man-Computer Communications Society,
Waterloo, Ontario, Canada
June 10-12, 1981.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-78-C-1551. It was also partially supported by the Office of Naval Research under contract N00014-76-0874, and by a grant from the Palo Alto Research Center of Xerox Corporation.

Table of Contents



List of Figures

Abstract

ZOG is a rapid-response menu-selection system whose databases are networks of frames or screenfuls. ZOG's frame editor, ZED, combines facilities like those of other text editors, and facilities specialized to the network character of the ZOG database. This paper describes an effort to evaluate ZED for overall ease of use, time to do specific tasks, and keystrokes required to do a task.

We began with Teresa Roberts' study, *Evaluation of Computer Text Editors*, which uses specific document editing tasks to compare four text editors -- Wang, NLS, TECO, and Wylbur. Four subjects did Roberts' tasks using ZOG. The results showed that overall task time in ZED was most comparable with TECO, Roberts' slowest editor.

To analyze the sources of the task time in detail, we applied Card, Moran, and Newell's keystroke level analysis of search-modify-verify cycles in editing (*Keystroke-Level Model for User Performance Time with Interactive Systems*). Using this method, we were able to account for most of the actual time required to do the tasks. This level of analysis and the attendant partitioning of the total editing time showed specific types of tasks at which ZED excelled and others at which ZED was less efficient than the other editors. We thus identified specific system changes which would result in immediate and visible improvements in editing with ZED.

1. Introduction

In the past few years there has been a growing interest in evaluating human-computer interfaces, including interfaces to computer text editors. Several studies (Card, Moran, and Newell, 1980a, 1980b) model users' interaction with an editor in terms of keystrokes and time required to acquire the next unit of text modification. Another study (Roberts, 1979) applied this model to compare experts' editing time on a standard task for four editors--TECO, Wylbur, NLS, and Wang.

ZOG is an experimental interactive system developed at CMU (Robertson, McCracken, Newell, 1980). One of our goals in developing ZOG is to find how to respond rapidly to user difficulties by effective diagnosis and subsequent system modification. To this end we are searching for methods of evaluating systems undergoing frequent design changes. ZOG contains an editor, ZED, which combines facilities like those of standard text editors with facilites specialized to the network character of ZOG's data base (McCracken and Robertson, 1979). ZED is becoming increasingly important, but was not initially the major focus of ZOG's design. The keystroke model and Roberts' "prepackaged" editor evaluation offers the possibility of comparing ZOG/ZED with other systems. Conversely, ZOG/ZED offers an interesting test of whether these techniques could be extended to a somewhat different system.

This paper presents the initial results of such an evaluation. Below, we first present a brief description of ZOG. Following this, we describe the experiment we performed. Then we discuss the results of comparing ZOG with TECO, Wylbur, NLS, and Wang, and the results of our keystroke level analysis of ZOG use. Finally, we summarize system design changes suggested by these results.

1.1. What Is ZOG?

ZOG is a general purpose, rapid-response, menu-selection interface to a computer system. ZOG's databases are strongly hierarchical, multiply linked *nets* of displays called *frames*, each the size of a terminal display screen. Each frame (see Figure 1) consists of a set of *items*: title, a few lines of text, a set of numbered (or lettered) menu items called *options* and *local pads*, and a line of ZOG commands called *global pads* at the bottom of the screen. Global pads include *back* (back up one frame) and *edit* (edit the current frame). An option, local pad, or global pad is selected by a single character, usually the first in its description.

An option or local pad can point to a program and/or another frame. Local pads usually point down subsidiary paths in the net. When the user makes a selection, the system executes the program or displays the appropriate next-frame. This structure allows rapid traversal of large amounts of information, with the system guiding the user in natural language. If the user selects an option or local pad with no next frame, ZOG will, at the user's option, create a new frame linked to that selection. ZOG then places the user at the new frame, in the editor (ZED). Thus a user creating a ZOG net moves freely between ZOG selection mode and ZED.

ZED is a display editor with a large set of commands for editing the textual content of the frame, rearranging the positions of items on the frame, and editing the non-displayed information such as next-frame links. Most ZED commands are single characters. After the user has selected *edit*, all keyboard input is interpreted as ZED commands rather than ZOG selections. Within ZED there are several modes: *command* mode, in which characters are interpreted as commands and command arguments, *insert* mode, in which characters are inserted into the text at the current cursor location, *position-item* mode, and ZED *help*. The *exit* command returns the user to ZOG selection mode.

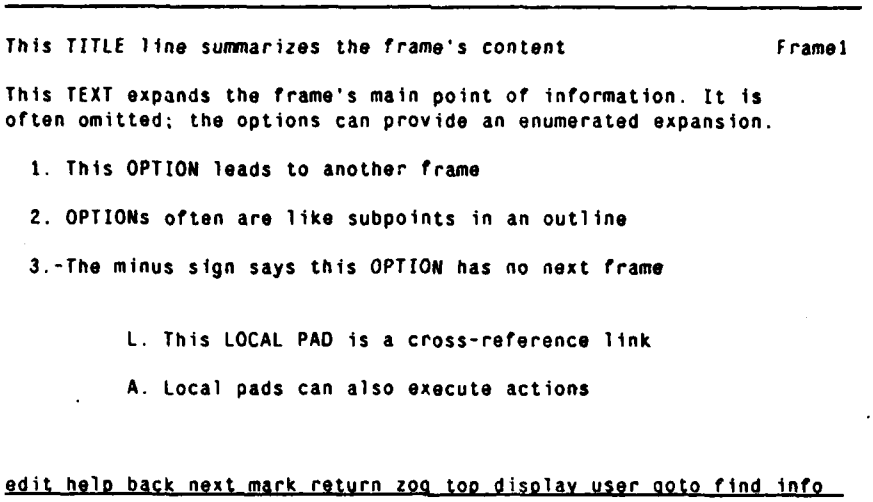


Figure 1: A Self-Describing, Typical ZOG Frame

2. Methodology for the Experiment

In describing our experiment, we will use terminology consistent with the model of editing of Roberts and Card, Moran, and Newell, who conceptualize editing tasks as follows. The user breaks his editing into units called *unit tasks*. Each unit task consists of (1) *task acquisition* (reading manuscript or screen), and (2) *execution* (searching for the text to be modified and modifying it). There may also be some *verification*. The sequence of edit commands used in execution is called a *method*. Basically, a unit task is the set of text modifications which the user can absorb and accomplish with one substantial look at the document.

2.1. Roberts' Methodology

Roberts developed a set of experiments including tests of experts' use of a set of commonly used *core* commands, and tests of functionality, ease of learning, and error and disaster potential of an editor. For this experiment, we used only the 53 editing tasks of the "expert core speed" experiment. These tasks are contained in a set of six short documents each marked in red with corrections the user is to make. If the user omits a task or does a task incorrectly, he must go back and make corrections at the end. Data are collected from editing of the last four documents: the first two are for practice. A typing test, administered at the end of the tasks, is provided to calibrate for the user's typing speed and accuracy. The principal data collected are: (1) *total task time*, including *real time* to edit the four documents plus *correction time* at the end; and (2) *error-free time* derived from (1) and a log of errors and corrections. Error-free time includes system delays and acquisition time. The total time and the error-free time are to be compared with those of other editors. Using this procedure, we hoped to compare ZOG/ZED with the editors Roberts studied.

2.2. The Keystroke Model

The keystroke model provides a way to characterize editing methods and the expenditures of time during editing, in detail. According to the model, the time T to do an editing task is composed of T_{acq} (acquisition time) and T_{ex} (execution time):

$$T = T_{acq} + T_{ex}$$

T_{ex} is the sum of system response times and time spent making keystrokes, drawing, or homing to a pointing device or keyboard, plus a mental preparation time for each action executed. As we apply it to ZOG use, execution time is:

$$T_{ex} = T_{keystrokes} + T_{mental\ prep}$$

The model posits a time for each operation: the user's typing rate for keystrokes, and 1.35 seconds for mental preparation. For an expert user, a mental preparation time is assumed for any command or keystroke which was not completely determined by previous actions. System response time counts only if it causes the user to wait; we were able to factor out these perceptible delays empirically.

The significance of this model is that if it does in fact apply to a given system, and if we can factor out system delays, we should be able to predict the amount of time an editing task will take based solely on the user's keystrokes. Thus if we find it does apply to ZOG use, we can use it to predict the duration of ZED editing tasks. With this tool, we can gather insight into the nature of ZOG use, and we can predict time savings resulting from specific improvements in ZED before we change the system. Since Roberts also collected some keystroke data, this approach provides a comparison in addition to task time, with her editors.

3. Procedure

3.1. Users

Users were five computer scientists, each with several years' experience in computing and in text editing. The users had each used ZED for over a year and were considered expert in its use. Data from user S2 were not used because he exhibited a variety of behaviors not exhibited by the other users.

3.2. The Task

We mapped Roberts' documents onto ZOG frames, one paragraph per frame where possible, with additional frames indexing the sections and paragraphs of each document. Mapping the documents onto frames was not the obvious procedure we anticipated, and the mapping we chose resulted in some difficulties during the experiment. ZOG frames with their titles and options have an appearance which can make it difficult to match a frame against a manuscript with the same content. Also, our duplication of the first lines of paragraphs as option texts in index frames created a problem: if the option pointed to a frame whose first line was to be edited, the user assumed he was to edit the option text as well. Editing of option texts was later deleted from the data; these unit tasks are designated *artifact tasks*.

A copy of the document net was created for each user to modify. One user at a time sat at a Minibee terminal with a 9600 baud hardwired line to a DEC Vax 11/780 computer. ZOG was already invoked; the user saw a list of selections, each leading to one of the documents he was to edit, and in the order in which they were assigned. He could ask questions about the tasks at any time.

3.3. Data Collection

Each user was filmed on videotape. A copy of the screen text the user was reading was superimposed on the television picture, along with a millisecond timestamp. Videotape data were accurate to one thirtieth of a second (the frequency of the video frames). During the session, ZOG unobtrusively recorded the user's path through the net and the selections and editing commands at each frame, each timestamped, on a log file. These data were pooled to identify errors and to partition the total task time among delays, acquisition time, and task execution time.

3.4. Treatment of Data

Actual editing methods were obtained for each user from the log files. Acquisition intervals were logged by scanning the videotapes and recording the appropriate time stamps. Acquisition was considered terminated if the user's head turned away from what he was reading, or if he struck a key. The portion of the acquisition time which occurred while ZED's "Edit" flag was on the screen was assigned to the ZED partition of the user's time, and other acquisition time was assigned to another partition (composed of searching the net or other non-editing activities in ZOG). The latter partition will be called *net behavior*. ZED and net behavior partitions were also obtained for non-task time, errors, and system delays. Delays while entering or exiting ZED were also logged.

To obtain the keystroke predictions, we at first used the users' typing rates as recorded from their typing tests. However, since our initial keystroke predictions were very low, we analyzed all individual *inserts* (strings of text inserted while editing). Calculating typing rate from *inserts* in the same way as we had from the typing tests, we found that actual typing during the tasks was much slower (average: .32 seconds per character) than during the typing tests (average: .20). Therefore each user's *insert* typing rate was used in the keystroke prediction of the time to execute his editing methods.

Finally, based on our users' distribution of acquisition time, we modeled their unit task behavior as: (1) one net behavior unit task to get to a target frame to edit; (2) one unit task composed of entering ZED and editing until the user went back to the document for another piece of his editing assignment; and (3) additional unit tasks within ZED. Finally, since exiting from ZED took a noticeable amount of time, users tended to look back at the manuscript after the delay to acquire the next task. Thus (4) exits were counted as unit tasks unto themselves.

3.5. Iterations

Our users S1 and S3 exhibited a clear need for some changes in ZED: exit times were inordinately long, and edit commands for finding a string (rather than just a single character) and for moving text within a frame were needed. We therefore improved exit speed and added the two editing capabilities. Some individual ZED operations were also made faster. As stated above, one of our goals was to find ways to evaluate a changing system; if the keystroke model proved applicable, we could use it to verify the task time improvement from these changes. S4 and S5 did the tasks with the improved system.

4. Results of Comparison with Roberts' Editors

4.1. ZOG Task Times

Figure 2 partitions ZOG total time to edit Roberts' documents. The first four columns represent our four users. The fifth contains the averages for the four users. The remaining columns contain Roberts' average times for TECO, Wylbur, NLS, and Wang, converted to seconds. Her figures are adapted from her Table 3.2 (Roberts, 1979).

ZOG total time, comparable to Roberts' total task time, includes: correction time, "non-task" time (such as clarifying correction marks); error time; artifact task time; ZED enter/exit delay time; other specific system delays (for such things as redisplay after justifying frame text); acquisition time; and actual task execution time traversing the ZOG net and executing ZED commands. For purposes of discussion, acquisition time is partitioned by net behavior and ZED editing.

To be comparable to Roberts' error-free time, ZOG figures must not include error time or any time not actually spent on the task. However, in Roberts' accounting of unit tasks, there is always one enter and one exit per document. The enter and exit commands themselves are a legitimate part of ZOG editing. However, the large enter and exit delays do not correspond to anything in Roberts' data. Also, improving the enter and exit delays was the major system modification between our two pairs of users (S1, S3 average: 474 seconds; S4, S5: 167). Since the purpose is to allow direct comparison of actual unit task times, we also subtract enter and exit delay time to obtain our error-free time. Other system delays remain in ZOG error-free time.

4.2. Error-Free Time Results

The average ZOG error-free time, 2766 seconds, is clearly at the time-costly end of the spectrum of editors. However, ZOG/ZED, TECO, and Wylbur are difficult to separate clearly due to the small sample sizes. A t-test ($\alpha = .02$) shows that neither ZOG and TECO, nor TECO and Wylbur, are significantly different, but that ZOG and Wylbur are. (Roberts showed that TECO and Wylbur are significantly different from NLS and Wang.) Nevertheless, it is clear that much about ZOG/ZED must be analyzed and can be improved.

4.3. Unit Task Comparison

As stated above, a unit task takes one of three forms: (1) net behavior; (2) editing, sometimes combined with entering ZED; or (3) exiting ZED. On average, ZOG users did the entire set of tasks in 142 unit tasks. The table below shows that ZOG users performed only 30% to 40% of their total unit tasks within ZED. They did 26% more unit tasks than TECO users (the one data point provided by Roberts).

	S1	S3	S4	S5	Ave	TECO
Net	39	50	51	49	47	
ZED	47	45	54	59	51	
Exit	39	48	46	41	44	
Total	125	143	151	149	142	113

These figures allow us to calculate the average task acquisition time for unit tasks involving text modification--a figure which can then be compared with similar figures for previous editing studies. For our users, the average acquisition time per ZED unit task was 7.0 seconds.

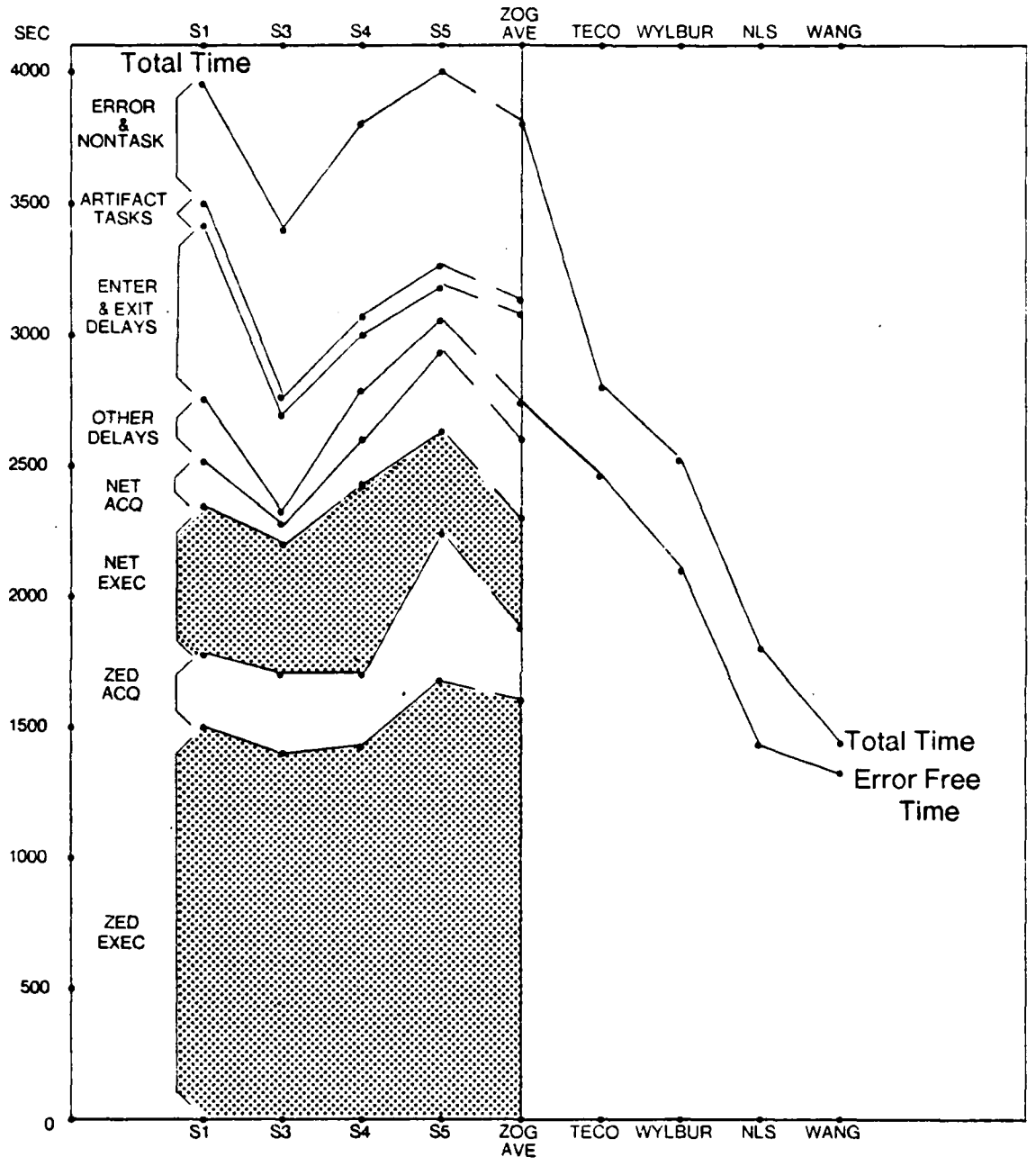


Figure 2: Real Task Times

5. Discussion

5.1. Mapping as a Source of Error-Free Time

Since the amount of error-free time spent by ZOG users is comparatively high, we would like to understand the sources of the extra time. One strong candidate is an artifact of the task. We have mapped linear text onto hierarchically structured frames to compare editors. On frames containing paragraph text, the frame title, frame name, and selections influence the overall shape of what is on the screen. Further, the manuscript contains nothing corresponding to the index frames. The user must verify that the first lines of paragraphs (the frame's option texts) are the appropriate ones for the path he is taking. In fact, our users spent about 50% of their net behavior acquisition time looking rapidly from screen to manuscript and back. Only about 20% of their ZED acquisition time was spent in rapid comparison. This indicates that during net behavior, the user is spending some effort locating himself in the net as opposed to deciding what to do for his next task. Also, to go to a location "lower" in the manuscript, the user may have to go "up" in the hierarchy of frames and then down another branch. This operation may require additional mental calculation.

We can also look at ZOG's large error-free time in terms of the unit task results. ZOG users' average number of unit tasks is noticeably higher than that of Roberts' TECO users. In the methods which Roberts predicted her users would use, one unit task generally corresponds to the editing changes at one location in the manuscript. One of her tasks contains (in general) one short search per location.

This contrasts with typical ZOG use, where a net unit task--an extra search--is usually required to locate the target frame. As Figure 2 shows, out of roughly 2000 seconds of execution time, about 25% is spent searching for the frame to edit. Including the exits, the ZOG user does at least three unit tasks for every time he enters ZED. The more steps there are in finding the location of the next text to modify, the more search and verification of location there is likely to be before any editing is done. Thus part of the time and many of the unit tasks may be artifacts of mapping linear text onto hierarchical frames.

5.2. Acquisition Time

The 7.0 second average ZED unit task acquisition time is somewhat longer than we would expect from Card, Moran, and Newell's results of 4.0 plus or minus 1.9 seconds. The most likely source of extra "acquisition" time is the fact that it was difficult, empirically, to separate any verification time from acquisition of the next task. Their studies effectively eliminated verification time. For comparison, according to Card, Moran, and Newell, acquisition takes only 1.8 seconds per unit task if scanning the screen had not been required (as with nondisplay editors).

5.3. Conclusions from Editor Comparison

The mapping problem may indicate that Roberts' stimulus must be adjusted for use with a net structured system. We could have avoided certain problems (the artifact tasks) by mapping paragraphs onto frames differently, but the user would still have to "translate" paragraph format into frame format as he searches for the frame to edit. For ZOG users, it might be appropriate to format the hardcopy document containing the 53 tasks to look like a set of frames.

Nevertheless, Roberts' method was beneficial in two ways. First, in partitioning system delays we identified several specific sources of delay which should be improved, including exit time and certain ZED operations such as *justify*. Along this line, in calculating the *insert* typing rates, we noted that 7

to 12% of all *insert* time was spent making typographical errors and correcting them with individual character-erase commands. This suggests that an erase-previous-word command would make editing easier. Second, the numbers of unit tasks and keystrokes, as well as error-free time, indicate that ZED users are spending more time than Wylbur, NLS, and Wang users. This indicates that we should look more closely at system changes which would improve overall task time in ZED.

6. Keystroke Level Analysis

6.1. Results of Keystroke Analysis

Figure 3 shows the result of our keystroke model prediction of task time, based on each users' actual methods and *insert* typing rate. The points on the graph are theoretical task times calculated by the execution time formula discussed above. The shaded areas represent the unpredicted time for net behavior and for ZED. On average, the new predictions account for 88% of ZED execution time, but for only 49% of net behavior execution time.

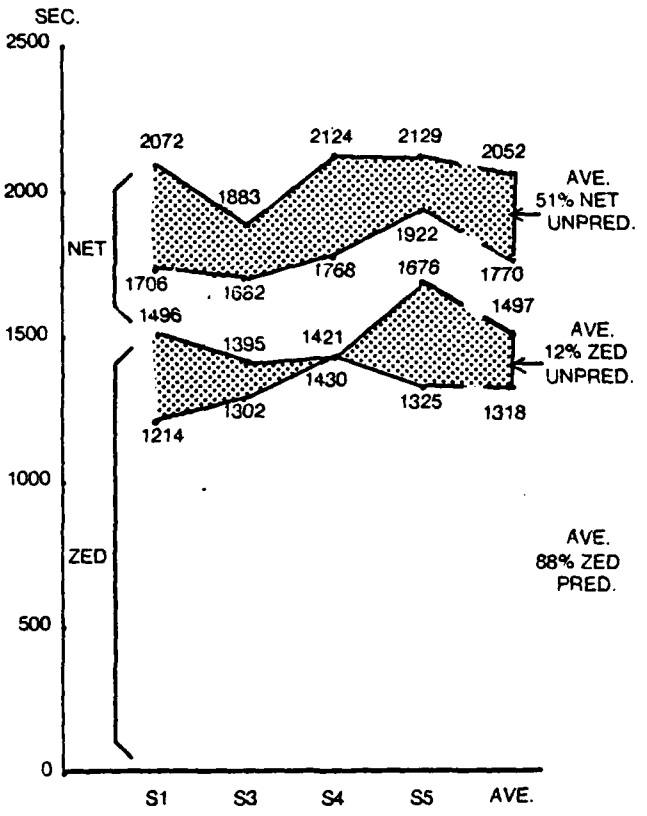


Figure 3: Predictions for ZOG Execution Time

6.2. Discussion of ZED Keystroke Predictions

These results show that the keystroke model in its present form cannot tell us very much about net behavior. Further experiments will be needed to identify the type of adjustment which would allow us to use the model to describe this aspect of ZOG behavior. However, we have accounted for most of the ZED execution time with the keystroke prediction. The question is, have we modeled ZED behavior well enough to warrant using the model for further analysis of ZED? The original experiments (Card, Moran, and Newell, 1980b) verifying the keystroke model (to a root mean square error of 20% on a single unit task) indicate that the the model should predict 97% of the real execution time for an experiment with 53 unit tasks. Thus our prediction error of about 12% indicates that we have not accounted for some aspect of the users' behavior.

The videotapes suggest one possible source for the unpredicted task time in ZED: the possibility of an extra mental calculation involved in switching levels or contexts within ZED. Since a frame is a collection of somewhat independent items (text, title, selections), there are two levels of cursor-moving commands (within, and between items). Besides keeping track of the item he is editing, the user must remember several contexts: net search versus edit, and within edit, normal search/modify context versus insert, position-item, or help. Although ZED experts move among the contexts and items rapidly, perhaps they would be modeled more closely by incorporating an additional mental preparation for every context change. If so, this suggests one source of small and highly distributed delays which may contribute to ZED execution time.

This type of discrepancy does not necessarily mean that we cannot make use of our predictions, however. In this regard, it is useful to compare our results with Roberts'. First, we note that Roberts predicted the editing methods her users would use, and predicted the task time from these methods. Her users' actual methods were somewhat more cautious (using longer search strings, for example). Also, her predicted times include a predicted amount of task acquisition and a small amount of system delay, which ours do not include. Thus her predictions are not exactly comparable with ours (which are based on empirical methods and do not include acquisition or delays). Nevertheless, given the aspects of the tasks for which she made predictions, her results are a useful yardstick for ours.

Comparing predicted time for predicted methods against real time to use those methods, her prediction error was about 30% overall. Her conclusion is that, with the caution that predicted times must be multiplied by some factor to represent real time, the prediction can be used to describe editing behavior. Based on this argument, our prediction error is low enough to allow us to use the keystroke model to analyze most of the time expenditure in ZED behavior.

7. Application of the Keystroke Model

7.1. Original ZED vs. Theoretical Minimum Time

Our first application of the keystroke model apart from modeling empirical behavior is to find a lower bound for ZED execution time. We constructed a set of editing methods for doing the 53 tasks, based on commonly used commands and command sequences available in the original version of ZED. The methods were constructed so that, based on keystroke model predictions, the tasks would collectively take the minimum possible execution time. Besides showing the theoretical minimum time for ZED, this gives us a way of estimating the amount of execution time due to users' choice of methods. Figure 4 gives the results of this prediction; the first bar, labelled "ZED1," refers to the original version of ZED. The bottom (shaded) portion of the bar is the keystroke prediction. The entire bar including the white portion represents the average predicted ZED execution time for the

actual methods of S1 and S3, who used this version of ZED. Our users' average *insert* typing rate, .32 seconds per character, was used as the keystroke duration in this and subsequent time predictions for minimum-time methods. This prediction indicates that our users could, theoretically, have accomplished the task in 23% less time. Possible reasons for their large empirical task times will be discussed in the next section.

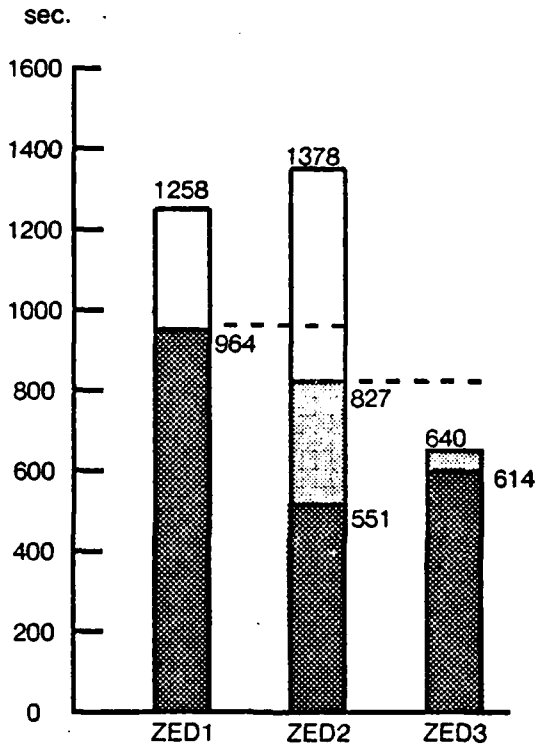


Figure 4: Predicted Execution Times--Three Versions of ZED

7.2. ZED1 vs. ZED2

Next, we compare the original ZED with the improved version. As stated above (see Section 3.5), between S3's and S4's sessions, we improved the speed of ZED exits, and added *find<string>* and a facility for copying or moving any text within a frame. The combined shaded portions of the second bar (ZED2) in Figure 4 represent the minimum-time prediction for this second version. The lighter shaded portion of the bar indicates the amount of the execution time predicted for all unit tasks directly affected by the system change. The entire bar including the white portion represents the average predicted ZED execution time for the actual methods of S4 and S5, who used this version of ZED. The space between the shaded area of the ZED2 bar and the dashed line is the time saving resulting from the design change. (The third bar, partitioned similarly to the ZED2 bar, represents a hypothetical ZED3 which will be discussed below.)

Thus we see that (1) users using both versions of ZED could have spent less time executing the ZED unit tasks, and (2) theoretically, ZED2 allows some time saving over ZED1. Taking this second observation first, the graph shows that in going from ZED1 to ZED2, the minimum execution time decreases by 14% of predicted ZED1 time. This is equivalent to 33% of the predicted time associated with unit tasks affected by the system changes. If we incorporate system delays, we would see additional improvement (since most of the actual system changes had to do with delays).

We can now address the question of the difference between the minimum execution times and the users' predicted times. One source of the difference can be seen by comparing the users' specific methods with corresponding minimum-time methods. As in Roberts' study, the users actually used more commands and longer *find* strings than necessary. For example, each user tends to use command strings such as *k.k.k.d* (*kill to next period* three times and *delete the next character*, to delete three sentences). The user could have typed *3k.d*. Similarly, we observe whole series of *s<c>* (search for single character *c*) when the user could have used *find<string>*. One explanation is that *3k.* and *find<best-string>* require extra calculation. With a display editor, and a set of single-character commands with mostly single- or zero-character arguments, it is easier to wait for the redisplay and react to it than to be sure ahead of time where the cursor will come to rest.

Now consider why ZED1 users' time was smaller than that of ZED2 users. In fact, S1 and S3 largely avoided the ZED commands which were subsequently made faster. Thus we would not necessarily expect to see a decrease in real time between pre- and post-system improvement users. That S4 and S5 actually took more time may be a matter of individual differences. However, their time is due in part to the invention of methods not anticipated by the designers. For example, they used the new *find* command to ask ZED whether a given string was on the current frame or not. This saved reading a long frame text and may in fact have substituted for additional acquisition time. In task execution, this method appears as a set of commands with no counterpart in the methods of S1 and S3. Further analysis of the relationships between predicted and actual editing methods will require more data.

However, in the absence of large amounts of data on users and with large differences in editing style, the ZED1-ZED2 predictions allows us to make a preliminary assessment of our design changes. Our users later said that they liked the new commands, and the commands became a permanent part of the ZOG system. As noted above, enter-exit delays, improved in this iteration, did drop off from S1 to S5.

7.3. Prediction for a Future Iteration--ZED3

Our final keystroke model analysis predicts the results of a hypothetical system version, ZED3 (see Figure 4). One clear deficiency in ZED is the lack of facilities for moving text or whole items between frames. ZED3 will have this capability, as follows. ZED2 saves in a buffer whatever text has been deleted from an item. The contents of the buffer can be cleared with the *clear* command and retrieved with the *caret* command until the next exit from ZED. In ZED3, the save-text buffer will not be cleared at an exit but will save the latest deletions to some maximum length (losing the earlier deletions as necessary). This allows the buffer's contents to be inserted on another frame. ZED3 will also have an analogous set of functions and a buffer to save any whole frame item which has been deleted. For selections, this means saving the one-line option text plus its designated selection character, any action triggered by the option, and the next-frame pointer associated with the option. Since the two buffers are saved continuously, the user must clear them before deleting any text or item he wishes to move.

In Figure 4, the ZED3 bar shows that the above design change would save 23% of the ZED

execution time of ZED2. This is equivalent to 88% of time involved in unit tasks affected by the changes. The reason for the large saving is that the system changes save typing several hundred text insert characters, a time-costly and error-prone activity. (In general, error-time savings do not appear in the execution time prediction; however, fewer typing errors would appear in the form of a lower average keystroke duration when the keystroke model is applied.) Examination of our users' actual methods shows that there would be a real saving. (For example, 12% of S5's predicted ZED execution time is saved if these new methods are substituted for his corresponding methods.) Thus we see that the keystroke model can be a useful tool in evaluating current and future system iterations.

8. Conclusions

Our study has shown that both Roberts' methodology and the keystroke model can be applied to practical evaluation and improvement of an interactive system. Despite ZOG/ZED being quite different from standard editors, we have found a way to make useful comparisons. The two techniques are complementary, and when combined with an iterative approach to system development, are an inexpensive way to analyze design changes.

ZOG's strong point remains its hierarchical structure. Moving or deleting large parts of the net is easy because of the modular nature of frames. This structure also made traversing large parts of a document relatively easy. However, analyzing the nature of net behavior remains for future studies. Future studies should also investigate how much of the execution time is due to the format of the manuscript from which editing is being done.

During the course of the experiment, we identified and tested several improvements, specifically shorter exit delays and several potentially time and effort saving ZED commands such as *find<string>* and the copy/move text facility. The experiment also identified several improvements for future system versions, specifically the ability to move or copy text and items between frames. A backward-erase command in ZED and additional speed improvements would also produce visible time savings. We have also identified context calculation in ZED and strings of short, uncalculated commands as sources of unpredicted execution time. However, ZED's weakest point appears to be the many enter and exit transitions between net searching and editing. Perhaps a future version of ZED could execute *find<string>* over multiple frames, do a depth-first search of the net, and leave the user editing the frame containing *<string>*. However this problem is solved, we will continue to consider new user problems and new design changes. Future system versions combined with theoretical predictions of the improvement should greatly increase our ability to improve ZOG.

Acknowledgements

The authors are grateful to George Robertson for contributions to all aspects of this work, and to Sandra Esch for many hours of data analysis and for drawing the figures for this paper.

References

Card, S., Moran, T., and Newell, A. Computer Text Editing: An Information-Processing Analysis of a Routine Cognitive Skill. *Cognitive Psychology*, 1980a, *12(1)*, 32-74.

Card, S., Moran, T., and Newell, A. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, 1980b, *7*, 396-410.

McCracken, D., and Robertson, G. Editing Tools for ZOG, a Highly Interactive Man-Machine Interface. In *ICC '79 Conference Record*, : IEEE Communications Society, 1979.

Roberts, T. *Evaluation of Computer Text Editors.* Technical Report SSL-79-9, Xerox Palo Alto Research Center, November 1979.

Robertson, G., McCracken, D., and Newell, A. The ZOG Approach to Man-Machine Communication. *International Journal of Man-Machine Studies*, 1980, in press.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CMU-CS-81-112 — 2. GOVT ACCESSION NO.: AD-A104 827	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): EXPERIMENTAL EVALUATION OF THE ZOG FRAME EDITOR	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): C. Kamila Robertson, Donald L. McCracken, Allen Newell	8. CONTRACT OR GRANT NUMBER(s): N00014-76-0874
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA. 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: April 1981
	13. NUMBER OF PAGES: 18
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)	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): Approved for public release; distribution unlimited	
18. SUPPLEMENTARY NOTES	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)	

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)

403 0x1


DATE
ILME