THE DEVELOPMENT AND VALIDATION OF A USER'S MODEL FOR INTERACTIVE TEXT EDITING

by

Lisa Joanne Folley

Thesis Proposal submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
Industrial Engineering and Operations Research

APPROVED:

R. C. Williges, Chairman

R. W. Ehrich

J. S. Greenstein

May, 1982
Blacksburg, Virginia

ACKNOWLEDGEMENTS

This thesis work was sponsored by the Office of Naval Research under contract N00014-81-K-0143, Work Unit number SRO-101. Dr. John O'Hare served as contract monitor. The opinions expressed in this thesis, however, are those of the author and not intended to represent official Navy views.

The author wishes to thank Dr. Robert C. Williges for his help in the planning and execution of this thesis work. Without his wisdom and guidance, none of this would have been possible.

Also due thanks are thesis committee members Dr. Joel S. Greenstein for his helpful comments and Dr. Roger W. Ehrich for the use of his software. James A. Pittman and Andrew M. Cohill were of invaluable help in using the software and computer systems. The assistance of Rich Critz and Joe Maynard in the transference of data to magnetic tape is also appreciated. The author would also like to thank Betty Chao for her preliminary work on this topic, and Dr. M. Lentner for his assistance in the interpretation of the cluster output.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Interactive text editors are an integral part of today's computer systems. They are essential to any computer system where files are created and updated. In spite of the prevalence of such systems and the importance of the interactive editor to the system, empirical research on the human engineering of such editors is lacking. Instead of being based on empirical evidence, the design of these editors may be based on intuition or convention. As Thomas and Carroll (1981) state, "...what is useful for the user is not [always] natural for the designer." (p. 247)

There is a definite need for an empirical data base to use as a foundation for interactive editor design and evaluation. For example, how many commands should be used? What functions should be included and how should those functions be arranged with respect to one another within the system? The design of the interactive editor can affect not only the amount of editing time and errors per editing session, but also user satisfaction and learning rate. For best user performance and highest satisfaction, editors must be designed from the user's point of view---as empirically

defined. The large number of new users being exposed to interactive systems adds particular emphasis to the need for human engineering in this area.

Empirical Research

The naming of edit commands is one aspect of interactive editor design which has received some attention. Ledgard and Whiteside (1980) found that simply changing command syntax to English form from a somewhat cryptic notational form improved performance considerably. Performance changes using the editor with English-like commands included more tasks completed, lower error rate, and overall higher editing efficiency.

Jackson and Tschirgi (1981) investigated novice and expert command naming. Each subject was led through a sequence of information retrieval or editing tasks. At each step, the subjects were shown the computer display of the operation, given the function description and asked to come up with a command name. The majority of the resulting commands were of simple structure with one verb and no dependent clauses. Overall, the commands were short, but the novices tended to be more verbose than the experts. Experts tended to respond with fewer inappropriate commands, and they assumed more defaults than did the novices. Perhaps

the most significant finding here is that there was very little agreement between or within the novice and expert groups on specific word choices. Therefore, any synonym list used by the computer to recognize commands would have to be very comprehensive---perhaps prohibitively so.

Dumais and Landauer (1981) performed two experiments which investigated the use of "natural language" for computer use. In the first experiment, subjects were asked to write a description of a word in such a way that 1) a computer ,or 2) a human, would be able to guess the word being described. An interesting result of this study is that when the word descriptions were intended for computers, novice subjects were more verbose and experts more terse. The two preferred forms of specification which came out of this experiment were 1) superordinate plus features--e.g., a list of keywords, with the first being the most important, and 2) lists of associates of the described word.

Similarly, the second experiment involved communication. Subjects were given texts with proofreader's marks and asked to prepare a list of instructions for someone else who would actually make the changes. For the most part, the commands given by the subjects did not agree with each other, or with the commands of the UNIX editor (a Bell Labs line-oriented

editor). The subjects tended to use the same name for more than one operation (for example, "omit" was popular for use on the word-level as well as line-level), which could pose an implementation problem. This second study also tested whether using the subjects' chosen command names helps people learning an interactive text editor. The results show that the time to perform the tasks was not significantly influenced by the command name set, and that there were few complaints about the inappropriate names used in the control condition. But these results may not hold with larger command sets or when long term recall is needed.

Thomas and Carroll (1981) performed a series of studies aimed at finding the most behaviorally effective names. As a result of these studies, they specify two important properties of command languages: congruence and hierarchy. In a congruent command language, commands which represent opposite functions are labelled with semantic opposites. For example, "release" and "take" are congruent command names, "unhook" and "grab" are not. Thomas and Carroll (1981) found that subjects who used a congruent command language learned more quickly and completely than those who did not.

Hierarchical command languages have multiple structural elements combined in different ways. For example, in Thomas

and Carroll's (1981) example of manipulating a robot, the commands "change arm open" and "change arm close" are hierarchical. "Change" represents a large class of categorization, "arm" is the part to be changed, and "open" and "close" are the exact actions to be performed. Hierarchical command languages were rated as better by subjects, were learned more quickly, and reduced the frequency of certain types of errors. Miller (1981) and Scapin (1981) have also explored the human factors of edit command naming. In general, the research seems to recommend the use of English-like syntax, and a command set which is congruent and hierarchical.

User Model Research

One approach to the human engineering of interactive editors involves the development of a user's model, or an empirical description of the user's expectations of the editor's operation. Since most editors have been developed with little regard to the user's model of the editing task, the user is often faced with the difficult task of determining the appropriate system model for that particular text editor. This affects not only novice users who have spent little or no time using any text editor, but also expert users learning an unfamiliar editor or implementing infre-

quently used commands. Models can be developed to describe or predict user behavior at the keystroke, command or general system level and then be used to tailor the design of interactive systems and software to user needs.

Keystroke Level Models. Some work has been done on model development at the keystroke level in interactive text editing. Simple models for use in predicting task time have been developed by Card, Moran, and Newell (1979) and Embley, Lan, Leinbaugh, and Nagy (1978). Predictive models such as these avoid direct measurement of editing time, which can easily be affected by variables such as editing command choice, user alertness and motivation, and error rate. According to Embley and Nagy (1981), keystroke models can predict task time for expert users on routine tasks (assuming zero error rate) with high accuracy.

One such model (Card et al., 1979) predicts task time by summing across unit task times.

$$T_{unit\ task} = T_{acquire} + T_{execute}$$

where $T_{acquire}$ represents time spent in mental organization and formulation of the task, and $T_{execute}$ consists of the sum of keying time, pointing time, homing time, system response time and mental response time. The validation of this model showed its overall task time prediction accurate to within five percent.

A similar model predicting task time was developed by Embley et al. (1978). In this model, unit tasks are considered single command-response pairs.

$$T_{task} = mT_c + nT_k, \text{ where}$$

m=number of command response pairs, T_c=delay per command, n=number of keystrokes, and T_k =time per keystroke. This model is a simple method for predicting task time for various command delays and typing rates.

Hammer (1981) has studied human performance on editing within a line. He developed a model to specify the optimal and constrained optimal number of keystrokes for a given user and edit change. He compared the actual number of keystrokes used to perform an editing change to the optimal and constrained optimal strategies. (Constrained optimal performance is defined as optimal performance limited by human information processing and motor capabilities.) Hammer found actual performance involved two to three times more keystrokes than optimal, and 1.5 times as many as constrained optimal.

The predictive power of these types of models, as stated by Embley and Nagy (1981), depends on:

1) how accurately quantities such as keystroke count,

typing rate, computer response time, and mental preparation are estimated; and

2) the validity of any simplifying assumptions.

Keystroke model predictions represent an upper bound on skilled user performance since they allow no time for editing method selection, and assume optimal command choices and flawless entry. Therefore, if large differences are found between predicted and observed performance, one may assume the editor is difficult to use optimally. However, keystroke models show how editing time is spent, but do not consider higher level processes.

Higher level models. The GOMS model, developed by Card, Moran, and Newell (1980) is an information processing model that describes how text editors are used to modify text. The GOMS model is flexible in that it can examine the editing process at different levels of detail. The level of detail, however, is not important in determining the model's power unless the sequence of editing operators can be predicted almost perfectly.

The GOMS model may be used not only to predict task time, but also method selection. The goals, operators and methods of expert users are developed by a few simple selection rules into a model which predicts the expert user's choice

among alternative methods for performing routine editing tasks. The model successfully predicted user method choice 80-90% of the time, predicted the sequences of operators 80-100% of the time, and predicted the time necessary to make modifications within 33% on new (crossvalidation) data and to within 4% on the whole manuscript editing task. The GOMS model is an example of a model which may be useful in the design and evaluation of interactive systems. Treu, as noted in Embley and Nagy (1981), has also gone beyond simple task time prediction. He has developed a model which predicts on a more specific level the mental work involved in text editing tasks. His predictions of detailed mental-time operators are based on action primitives and their relationship to system commands. Unfortunately he reports no empirical validation.

Another method of modelling involves a computerized file comparison algorithm (Embley and Nagy, 1981). The file versions before and after editing are compared to find matched, similar, and unmatched strings. The resulting comparison graphs can be used to derive the editing operations and generate a close to optimal edit command sequence. Editors can be compared in this way without the user biases of skill, experience, and preference. This approach can also be used to detect frequently occurring patterns of transformations,

and can produce models of types of editing activities, derive quantitative measures of editor quality, and lead to improved design guidelines. Its prediction, however, is only appropriate if the computer's editing strategy is based on the editing behavior of the humans it is modelling.

Roberts (1979) has studied text editing skill acquisition--how long it takes a beginner to perform basic operations such as insertion, deletion, replacement, moving, copying, splitting and merging. Roberts showed a significant difference among the four editors tested in terms of learning rate (number of tasks learned across time). Roberts also explored which editor features most affect the learning rate by counting the number of command types and items (i.e., command parts) necessary to learn the basic operations for each editor. She found that it is not the absolute number of command types that matters, but the number of items that must be strung together to perform a given editing task.

Roberts' methodology served also to evaluate the functional strengths of the whole system. Expert speed in using the editor also provided a measure for comparing the four editors. A further analysis--on the keystroke count level--could reveal <u>where</u> the time was spent. This can be useful

in deciding whether to use one command set over another, and whether the user speed is such that it would constitute a useful improvement to increase the hardware speed capabilities.

The approaches developed by Roberts can be used to make design judgements or to evaluate the different aspects of existing editors (as she did). For example, Robertson, McCracken, and Newell (1981) successfully implemented Roberts' methodology and the Keystroke Model developed by Card, Moran, and Newell (1979) to evaluate the ZOG frame editor.

Modelling by User Expectations. The models and editor evaluations mentioned above may be used in interactive editor design for creating less time-consuming and error-prone editors. Another approach might be to study only the new user's expectations of an interactive text editor's operation. For example, what does the new user feel command names should be? What operations does he or she expect to be present and how are they expected to work (i.e., what input is necessary for a given computer response)? In other words, what is the novice user's model of the editor? Performance on an editor based on this user's model should be superior to performance on another editor.

On the general system level, Mayer and Bayman (1981) have investigated user's models of calculator language and operation. They developed the user's models by asking the subjects to describe what they would see on the calculator display after certain numbers and symbols had been entered. Experts tended to have more sophisticated notions of the inner calculator workings than did novices. Specifically, they evaluated arithmetic expressions more as a calculator itself would. (For example, they more frequently evaluated multiplication before addition in a chain.) Mayer and Bayman suggest that cognitive objectives as well as behavioral objectives should be considered in developing user models. We should specify not only what we want the user to do, but also what we want him or her to know. Applying this to interactive editors, not only the user's actions, but also the user's cognitive model of the system should be considered.

Summary. Much of the empirical and modelling research has studied exclusively naive or expert editing behavior. A comparison between these two user's models would be most valuable, especially considering that both expert and naive users generally use the same editors. Assuming a user's model changes as the user becomes more experienced, what are the changes? What are the implications for editor design?

Is tailoring the editor to the user's experience level a possibility? The present study explores this naive/expert model comparison and its implications as well as the model development itself.

Approach

The approach in this research involves a two step process. The first step is the development of the user models, and the second step is their validation. This approach was recommended for several reasons.

First, the model development generates the expert and naive user's models using the somewhat artificial paper and pencil format instead of an actual interactive environment. If validation on interactive terminals is successful, the inexpensive and convenient paper and pencil procedure can be shown appropriate for use in simulating the actual interactive situation, at least for the purpose of model development.

Performing validation as a separate step also tests the models developed. The test takes place in the actual interactive environment and with subjects who did not participate in the model development study.

In the event that the models are not validated, the data from the validation study may, in turn, be used as the basis for new models--perhaps more appropriate because they will be generated in the interactive environment. Lack of validation of these models at some later date would argue against the clustering algorithm or other procedures used to generate the models. If, on the other hand, these models were eventually validated, the original lack of validity could be attributed to the use of the paper and pencil procedure.

MODEL DEVELOPMENT

Purpose

A paper-pencil procedure was used to develop user models of text editing. Both novice and expert user's models of interactive text editing were determined by empirically analyzing patterns of command language usage. These models, in turn, can form the basis for developing an interactive editor which is designed from the user, rather than the system, point of view.

Method

Command Names. Before the actual user model development was begun, a set of editing commands was compiled. Existing command sets were avoided to eliminate the possibility of any subject being familiar with the command set, thus biasing the experimental results. Care was taken to establish a natural, clearly defined command set since previous research suggests there is little agreement among users in terms of command names, whereas English-like command syntax and congruent and hierarchical commands can affect operator performance.

To arrive at command names clear to most users, a preference survey was conducted. Each of twelve naive subjects (no interactive terminal experience) and twelve expert subjects (at least 1000 hours on an interactive system) filled out a command name preference questionnaire. The questionnaire described 39 interactive editor operations and gave several appropriate command names for each. Subjects were instructed to read each description and either choose one of the listed command names or suggest their own.

The command names for each operation which received the most votes across all subjects were then chosen for inclusion in the editor. In the case of a tie, the shortest name was chosen. The resulting list of command names and corresponding functions which formed the "editor" used in the model development are presented in Appendix A.

Subjects. A total of 20 expert and 20 naive users of interactive computer systems were recruited from the university staff and student body to serve as subjects. Subjects were informed of their rights as subjects and asked to sign an informed consent form. (See Appendix B.) Naive users were defined as people who had never used an interactive terminal. Expert subjects had used interactive terminals for a median of 3 years and a median of 25 hours per week.

Other characteristics of the experts' computer experience are noted in Appendix C, the experience questionnaire completed by all expert subjects.

Task. Three editing tasks were presented to the subjects. Each of twenty different edit changes (such as delete word, copy line, insert line) was presented once in each of the three tasks. The twenty specific edit changes as listed in Table 1 were chosen to represent various editing tasks present at the character, word, line and general level. These edits were presented in a different random order in each task to balance any learning or fatigue effects.

Procedure. The naive subjects and the expert subjects participated in separate groups and each subject completed two sessions. The first session began with approximately one hour of instruction followed by one text editing task. All subjects read a one page introduction to the basic concepts of text editing which briefly described interactive terminals and text editing (adapted from Cohill, 1981). Next each subject received a copy of the complete list of editing commands and arguments and their functions used in the study, as well as a one page list of the command and argument names alone (no descriptions) for reference. (See

TABLE 1

The Twenty Edit Changes Contained in Each Edit Task

Character Level	Word Level	Line Level	General
Insert	Insert	Insert	Split
Delete	Delete	Delete	Join
Change	Change	Change	Append a string to the end of a line
Transpose	Copy	Copy	Delete to a string
	Switch	Move	Repeat the last edit change
			Perform the edit change on all occurrences of the pattern

Appendix A.) The subjects were given 20 minutes to study the command descriptions, after which the experimenter described, with the use of visual aids, the various editing operations and commands. Subjects were encouraged to refer to the list of command names and descriptions they had received, and questions were invited. The aim was to have subjects understand all commands so that they would use those they preferred rather than those they understood.

After the instruction was complete, the subjects performed Task 1 of the 3 editing tasks as practice. Subjects returned the following day for another session in which they performed the two remaining tasks. To eliminate possible order effects, half of the naive and half of the expert subjects received Task 2 first, the other half received Task 3 first. Each session lasted approximately 3 hours for the naive subjects and 2 hours for the experts. The actual text varied from task to task, as did the presentation order of the edit changes, but the general format and procedure were the same across all tasks.

The subject was given two copies of the text to be edited. A BEFORE text had 20 corrections marked in black with circles, arrows or underlines. Words such as "copy" or "delete" were avoided in order to prevent biasing the sub-

jects' choice of command. An AFTER text showed the manuscript in its corrected form. (Appendices E through J contain the three sets of BEFORE and AFTER texts.)

The subject's task consisted of listing the commands to be used in the order in which they would be used to make the given edit change. Each subject recorded his or her chosen commands separately for each task on an optical scan form for computer input. (See Appendix K for a sample answer sheet.) Subjects were asked not to erase commands they wanted to retract, but instead to add commands to correct any mistakes. In this way, the commands used should more closely simulate actual interactive terminal editing.

Results

Error checking. An initial analysis was performed to insure that subject responses were valid edit commands for the given editing task. The error rates were relatively low for both naive subjects (4.88%) and expert subjects (3.25%). These errors were of such small magnitude and, for the most part, inconsequential so no additional corrections were made. Once this error check was completed, the data were formatted into a 20 x 40 frequency matrix of edit tasks by edit commands for each subject.

Cluster analysis. The primary analysis of the data was a series of hierarchical cluster analyses conducted on the formatted 20 x 40 edit change by command frequency matrices to determine the expert and naive user's models. Cluster analysis was employed as a means of identifying the natural homogeneous groups, or clusters of the "items" under study. Through analysis of the user's behavior, the cluster method describes the user's structuring of the task. The experimenter then evaluates the cluster outputs to determine the number of clusters which best describe the homogeneous groupings of data. This subjective judgement may introduce some bias, but allows the selection of the number of clusters which form meaningful groups. These clusters then form a user model. Not only does this type of analysis describe the user's internal organization of the editing task, but also the actual methods used within that organization.

Clustering is particularly useful here since the matrix of data is so large and lacks obvious underlying structure. In fact, this underlying organization is what is being investigated. Because of the amount of data involved, simple inspection would be inadequate to discern the inherent groupings of similar items. Cluster analysis organized the data into clusters which represent certain empirically measured relations of similarity.

The clustering algorithm was chosen over other types of analyses for several reasons. The cluster analysis, modelling high level processes, supplies the means for specifying the user's organization of the editing task. Cluster analysis also supplies a specific command list which may be used in editor design. A keystroke model was not used because of the low level at which such models examine user behavior. Keystroke models include consideration of various time measurements, but do not provide a means of developing command set models. In general, keystroke level models are more useful in evaluating existing editors. In contrast, the cluster method approaches user modelling at the macro level of user conceptions of the editing task and not at the micro level of keystrokes. In the design of new editors, the models developed can prescribe not only how many, but which commands should be included for a given user or user group.

Models such as the GOMS model, although they do describe higher level processes, are also not as suitable as clustering in this application. The decision rules found for each user are generally based on frequencies of use of given strategies. Some method is then needed to group these strategies into homogeneous groups. If cluster analysis is used, this grouping is included in the analysis. In other words, the cluster method accomplishes more in terms of

developing overall user models which can then be used in editor design.

The clustering in this study was performed using the Statistical Analysis System , or SAS (1979). Use of the clustering algorithm gives rise to a kind of distance between the items being clustered. The resulting distance matrix is used to group the items into clusters.

$$D(u, v) = \sqrt{\sum_{i=1}^{n}(x_{iu} - x_{iv})^2},$$

where u is the u^{th} item,

v is the v^{th} item ($u \neq v$),

i is the number of categories across which items u and v appear,

x_{iu} is the number of times the u^{th} item appeared in the i^{th} category,

and x_{iv} is the number of times the v^{th} item appeared in the i^{th} category.

In the analysis of the present study, two such cluster analyses were performed separately on the naive and expert subjects' data. (For a further explanation of the clustering algorithm, see Appendix L.) First, the twenty editing changes (such as delete word, insert character, move lines)

were grouped, based on the commands used to perform them. The more similar the commands used to perform a given edit changes, the more likely those edit changes are to be in the same cluster. Second, the editing commands were clustered within each of these edit change clusters, yielding a core set of commands which formed the user model.

Edit change clusters. The cluster analyses across the two edit tasks were conducted both on each subject individually and on each subject group (naive and expert). In general, the individual subjects' models matched the group models. Of the 40 subjects' models, four individual user's models did not fit the group model---one naive and three expert. Due to this remarkably high agreement, only the overall naive and overall expert group data were considered.

When the naive group and expert group data were clustered across the twenty edit changes, the models of the naive and expert users were equivalent. As shown in the top portion of Figure 1, both groups yielded four general clusters. Edits were grouped into:

1) changing and inserting characters and words,
2) deleting characters, words and lines,
3) moving and copying lines, and
4) inserting and changing lines.

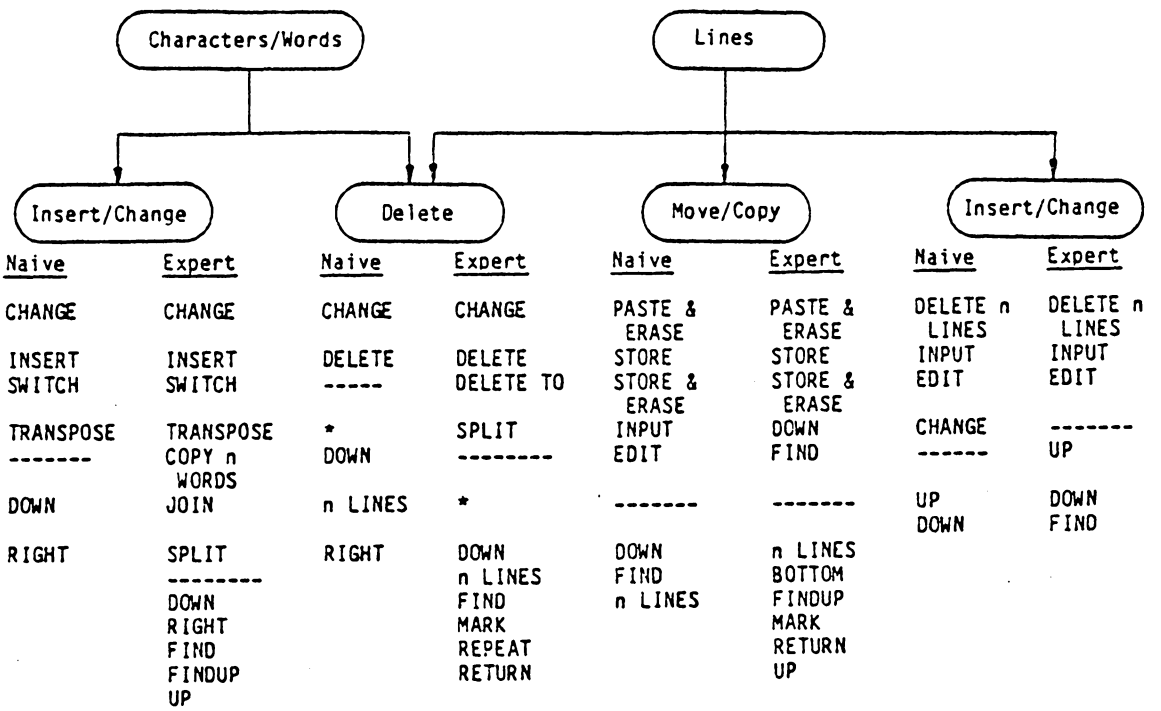


Figure 1: Summary of Naive and Expert User Command Clusters Under Each of the Four Edit Change Clusters

Command clusters. Each of the four edit task clusters was analyzed to determine the unique naive user's and expert user's clusters of commands. These cluster analyses essentially resulted in individual command clusters appropriate for each of the four general edit tasks and a large cluster of commands which were inappropriate for the particular edit task.

The commands which were appropriate for each of the four general edit tasks are listed separately for naive and expert users in the bottom portion of Figure 1 . Commands that are primary to each edit task are listed first followed by secondary commands (below the dashed lines in Figure 1) which are appropriate across several of the four general edit tasks. Clearly, the list of primary and secondary commands for the naive user's and expert user's clusters differ quite markedly, as shown in Figure 1, both across the four general edit tasks and in terms of the unique commands.

The set of 16 unique commands and arguments contained in the naive user's model is a subset of the expert user's model set of 26 commands and arguments, which is in turn a subset of the full set of 39 commands and arguments originally available to all subjects. Table 2 lists the commands included in the full editor, the naive model, and the expert

model. The expert user added some more powerful commands to the commands included in the naive user's model, but neither expert nor novice clusters incorporated the full set of available commands. Also in Table 2, under the appropriate model heading are listed the frequencies of each command's use for expert and naive subjects.

Discussion

According to the cluster analyses, both the naive and expert users agreed quite closely as to the four general edit tasks, but each group had a different user model for the command repertoire within each of the four general edit tasks. The naive user's model contains a smaller and simpler set of commands than does the expert user's model. This may be because the naive user's need to become familiar with basic text editing concepts decreases the available capacity for learning command names and functions. On the other hand, the naive user's lack of familiarity with any of the commands or functions could just make them more difficult to understand in comparison to the expert user's effort. The expert is not only already familiar with the basic concepts of interactive text editing, but may also recognize several command names or functions from work with previous editors.

TABLE 2

Summary of Frequency of Use of Unique Commands

Full	Expert	Freq	Naive	Freq
CHANGE	CHANGE	291	CHANGE	130
DELETE	DELETE	194	DELETE	298
DOWN	DOWN	695	DOWN	724
FIND	FIND	80	FIND	101
INSERT	INSERT	18	INSERT	149
PASTE AND ERASE	PASTE AND ERASE	72	PASTE AND ERASE	53
RIGHT	RIGHT	36	RIGHT	202
STORE	STORE	33	STORE	30
STORE AND ERASE	STORE AND ERASE	42	STORE AND ERASE	49
SWITCH	SWITCH	38	SWITCH	39
TRANSPOSE	TRANSPOSE	12	TRANSPOSE	38
UP	UP	40	UP	56
*	*	39	*	35
n LINES	n LINES	119	n LINES	131
INPUT	INPUT	89	INPUT	87
EDIT	EDIT	89	EDIT	84
BOTTOM	BOTTOM	15		
COPY	COPY	4		
DELETE TO	DELETE TO	35		
FINDUP	FINDUP	25		
JOIN	JOIN	47		
MARK	MARK	23		
REPEAT	REPEAT	44		
RETURN	RETURN	22		
SPLIT	SPLIT	48		
n WORDS	n WORDS	4		
LEFT				
TOP				
RTL				
#				
PASTE				
n				
DISPLAY STORE				
DISPLAY				
DELETE UP				
CLEAR STORE				
CANCEL				
AREA				
ADD				

Experts, as they gain experience, have also had a chance to develop more sophisticated editing strategies applicable across editors. These sophisticated strategies would likely include more powerful commands, i.e., commands that can perform the function of a sequence of two or more simpler commands in making an edit change. For example, in a string search, the naive user might use the DOWN command repeatedly, while the expert uses the more powerful FIND command. The combination of these factors can explain the fact that the expert user's model not only includes a greater number of commands, but is essentially the naive model set of commands plus some more powerful commands.

The model development study showed that the clustering analysis provided a useful method for inferring a user's model of the editing task. The models developed may be used as the basis for the design of an interactive editor designed from the user's rather than the system's point of view. An editor built around these key commands and edit task groups will conform to the user's idea of an interactive editor. As a result, the user should spend less time dealing with preconceptions incompatible with the editor and should show overall superior performance.

The truth of this hypothesis could be evaluated by comparing user performance on a model-derived editor to that on another editor. For example, a naive user should accomplish editing tasks faster and with fewer errors when using an editor based on the naive user's model than when using an editor not based on that model. Successful validation of the user's models would show that basing the development of an interactive editor on the empirically derived user's model does, in fact, lead to improved performance. The usefulness of the clustering algorithm as a means of model development would be demonstrated. Also, the paper-pencil editing format would be shown to be an effective way of avoiding excessive use of both terminal time and instructor/experimenter time.

VALIDATION STUDY

Purpose

The purpose of this study was to test not only the validity of the naive and expert user's models, but also of the paper-pencil format as a means of arriving at those models. The user model validity was tested by implementing interactive editors actually based on these models. The performance of subjects using these editors was then compared to the performance of subjects using standard text editors. It was expected that user performance on the appropriate model-based editor would be superior to user performance on another editor. Throughout the validation study, the format of the model development study was followed as closely as possible.

Method

Subjects. Eighty volunteers recruited from the university staff and student body took part in this study. All potential subjects passed a vision test on a Bausch and Lomb orthorater to screen out those potential subjects with visual acuity less than 20/75 or with severe phoria (binocu-

lar action of the eyes) problems. A typing test eliminated those potential subjects totally unfamiliar with the standard QWERTY keyboard used. Each participant was informed of his or her rights as a human subject and signed an informed consent form (Appendix B).

Experimental Design. The validation study was a 2x4x2x2 mixed-factor design. Figure 2 diagrams the experimental design. The factors of primary interest were the between-subject factors of user (naive or expert) and editor (full, expert model, naive model or control command set). Each subject qualified as either naive or expert and used only one of the four editors. The two control variables of text (two manuscripts) and practice (first or second session) were within subject variables--each subject edited both texts, one in each of the two editing sessions.

The two levels of user were naive and expert subjects, as determined by the user's previous interactive computer experience. A naive user was defined as someone who had never used an interactive computer terminal or interactive word processing equipment. Expert users had a minimum of 20 weeks of interactive terminal experience, either as part of school work or on the job. Table 3 contains a profile of expert subject experience.

		EDITOR							
		Full		Novice Model		Control		Expert Model	
		T1	T2	T1	T2	T1	T2	T1	T2
USER	N P1	1-5	6-10	11-15	16-20	21-25	26-30	31-35	36-40
	N P2	6-10	1-5	16-20	11-15	26-30	21-25	36-40	31-35
	E P1	41-45	46-50	51-55	56-60	61-65	66-70	71-75	76-80
	E P2	46-50	41-45	56-60	51-55	66-70	61-65	76-80	71-75

Note: 'P' represents the factor 'practice'.

'T' represents the factor 'text'.

'N' represents naive users.

'E' represents expert users.

The numbers in each of the 32 cells represent individual subjects.

Figure 2: Experimental Design---Validation Study

The four editors used were modified versions of SAM (Ehrich, 1981), an experimental editor developed especially for use in human-computer interaction studies. Included in SAM is a metering capability which was implemented to measure and record subject responses at the terminal. The following measures were included in this metering:

1) the commands used to perform each edit task;
2) the number of errors (i.e., unrecognized commands); and
3) total time to complete each task.

(The meter routines used with SAM were developed by R. W. Ehrich, and data compression software used in conjunction with the metering routines was developed by A. M. Cohill.)

Table 4 lists the complete set of commands for each of the four editors. (Corresponding commands are aligned.) The few minor changes which were needed are listed in Table 5 .

Six basic commands (AREA, CHANGE, DELETE, DOWN, UP, and CHANGE TO INPUT MODE) were included in all four editors. The largest of these editors was the full editor, which included essentially the full set of editing commands available in the editor used for the model development.

TABLE 3

Expert Subject Experience Questionnaire and Profile

---

1) How many school courses have you taken which involved a significant amount of interactive terminal use?

Median: 5 courses

Mean: 5.55 courses

Range: 0 to 12 courses

2) Have you had interactive terminal experience in addition to the above courses? If so, describe it.

Number of expert subjects with outside experience: 30 out of 40

3) When you edit interactively, do you edit text, data or computer programs?

Programs only: 17

Text only: 0

Data only: 1

A mix of the above: 22

4) With how many interactive editors are you familiar?

Median: 3 editors

Mean: 3.9 editors

Range: 1 to 16 editors

The three remaining editors are subsets of this full editor. Two of these, the expert model editor and the naive model editor were based on the cluster output of the model development study. The expert model editor contained those 21 commands which the experts included in their user's model. The naive model editor was based on the naive user's model, and is a 13 command subset of the expert model editor. The control editor, also a subset of the expert model editor, controls for editor size. Of its 13 commands, only six are in common with the naive model editor. The seven additional commands were the most frequently used powerful commands of the expert model editor.

The two final factors of text and practice served as control variables as shown in Figure 2 . The two texts used were the same as those in the model development study. The following minor changes were made to the hard copy BEFORE/AFTER texts used in the model development:

1) Margins were not adjusted in the AFTER texts. All lines remained as they would be after the specified edits were performed, resulting in some short and some long lines.

2) In BEFORE 2, all lines are double-spaced, as in the other BEFORE texts.

TABLE 4

Four Experimental Editors

Full Editor	Expert Model	Naive Model	Control Editor
AREA	AREA	AREA	AREA
CHANGE	CHANGE	CHANGE	CHANGE
DELETE	DELETE	DELETE	DELETE
DOWN	DOWN	DOWN	DOWN
UP	UP	UP	UP
INPUT	INPUT	INPUT	INPUT
FIND	FIND	FIND	
PASTE AND ERASE	PASTE AND ERASE	PASTE AND ERASE	
INSERT	INSERT	INSERT	
CUT	CUT	CUT	
CUT AND ERASE	CUT AND ERASE	CUT AND ERASE	
SWITCH	SWITCH	SWITCH	
TRANSPOSE	TRANSPOSE	TRANSPOSE	
DELETE TO	DELETE TO		DELETE TO
FIND UP	FIND UP		FIND UP
JOIN	JOIN		JOIN
MARK	MARK		MARK
RETURN	RETURN		RETURN
SPLIT	SPLIT		SPLIT
REPEAT	REPEAT		REPEAT
BOTTOM	BOTTOM		
TOP	TOP		
LINE			
ADD			
CLEAR STORE			
DELETE UP			
DISPLAY			
DISPLAY STORE			
PASTE			

Note: Identical commands are aligned across each row.

Each text contained the twenty edit changes (listed in Table 1) in different random order. This order and the different content of the two texts was the reason for their counterbalancing. Practice, a second control variable, was included to allow the isolation of any fatigue or learning effects which might be present.

Procedure. All editing sessions, as well as the typing tests, were conducted on DEC VT100 terminals. The terminals were connected to the main computer by dedicated phone lines using high speed modems which transmitted information at 9600 baud. The computer used was a VAX 11/780 with 2.25 megabytes of main memory. In addition, two DEC RM03 disk drives each held 80 megabytes of secondary storage.

Each subject was involved in two approximately two hour sessions which took place in a small room. The subject's work station consisted of a VT100 terminal and high intensity desk lamp on a terminal table 27" from the floor. Room luminance, measured from the wall at the work station by a spot photometer, was kept at 1.1 cd/m^2. Terminal screen luminance measured 2.2 cd/m^2. Within each of the two experience levels and their particular editor, the subjects received instruction individually or in pairs. Questions during instruction were encouraged and answered fully as

TABLE 5

Explanations for Command Changes

Commands and arguments removed:

COPY	Removed because redundant with other commands (e.g. PASTE, STORE on a line level, CHANGE, INSERT on a word level).
n WORDS	No longer necessary.
n LINES	No longer necessary.
CANCEL	Implementation would require rewriting the SAM editor.
n	Difficult implementation.
RTL	Difficult implementation.

Name changes:

LEFT	Replaced by backspace key on terminal keyboard.
RIGHT	Replaced by space bar on terminal keyboard.
*	Considered part of commands, because can be used with both change and delete.
INPUT	Replaced by null carriage return.
EDIT	Replaced by null carriage return.

old	new	
STORE	CUT	To maintain congruency with "PASTE".
STORE AND ERASE	CUT AND ERASE	To maintain congruency with "PASTE AND ERASE".
#	LINE	To keep all commands alphanumeric.

Commands added:

TOP	Added to expert model editor, since BOTTOM was included. Because the editing is interactive, TOP may be needed.
AREA	Added to naive model, expert model and control editor, so the subject can see where he or she is in the interactive file displayed on the terminal screen.

long as this did not bias the subject's command choice. Judgemental answers were avoided.

For session 1, each subject read the instructions of the model development study, paced by a tape recording of those instructions. Subjects were instructed to read along with the tape as it read 'The Basic Concepts of Interactive Text Editing' (see Appendix N) and went through a description of each command and its function. An example of the use of each command was included in the instruction manual, and described on the tape. See Appendix N for the instructions for the full editor condition. The total number of commands included in the instruction, of course, varied, depending on which editor the subject was being trained.

The subjects then performed a simple editing task which introduced them to the actual interactive use of the editor through a fading procedure. In this first editing, the subjects were given 'help' in performing the edit changes. On the manuscript hard copy were directions listing the commands to be used to perform each edit change. Appendix D contains this text (BEFORE 0). The BEFORE text, minus edit change marks and 'help', and single spaced, served as the computer file to be edited. In editing, subjects were permitted to use only the six basic commands common to all four

editors. As the subjects proceeded through the edit changes, the directions became less explicit until the last direction described only the edit change to be made, listing no commands. After completion of this task, the subjects were given a five minute break during which the experimenter set up the next task. The first of the three editing tasks of those used in the model development study was then performed as practice. (See Appendices E and F.) The subjects had no contact with each other while performing the experimental tasks. The experimenter answered any questions, as long as these did not bias the subject's choice of command.

The appropriate BEFORE text, without the underlines and correction marks, and single spaced, was presented on the computer for editing. The subjects had a hard copy of both the BEFORE text (with correction marks) and the AFTER text for reference as they edited. In addition, each subject had the list of command names with descriptions, and the one-page list of commands at hand for reference during task performance.

Session 2 took place after a 15 minute break. The subject performed the remaining two editing tasks which produced the actual analyzed data. These two tasks were separated by a five minute break, and their order was

counterbalanced within each condition. The BEFORE and AFTER versions of texts 2 and 3 are shown in Appendices G through J.

Immediately after each task, the edited file was compared to the actual AFTER text, i.e., to the way it should appear, by a computerized check. Any lines which differed between the two files were displayed for the experimenter. Any incomplete edit changes were pointed out to the subject on the hard copy BEFORE text. The subject then returned to the interactive editing until all edit changes were performed.

Results and Discussion

Three general analyses were performed on the validation study data. First a series of analysis of variance (ANOVA) tests examining the significance of the independent variable effects were conducted. ANOVAs on both overall and detailed dependent measures were evaluated. Second a test for validation of the clustering method used for the user model development was performed. Finally, frequency counts of command use were compared across the model development and validation studies, and across user and editor in the validation study.

ANOVA of overall dependent measures. The initial analysis of the data consisted of an ANOVA on several overall dependent variables. The dependent variables measured as performance indices included total time to complete the editing task (TT), total number of errors (TE), and total number of commands used (TC), summarized in Table 6 . TT was a measure of elapsed time from the start of the session until the subject entered 'stop' on the keyboard, when he or she was finished. If the subject had failed to make one or more of the 20 edit changes, the elapsed time of the subsequent 'fixing' session was included in the variable TT. The TE measure included syntactical but not semantic errors. In other words, commands and arguments not recognized by the editor, whether typing mistakes or simply incorrect commands, were included in the error count, but acceptable commands which did not perform the desired action were not. This was a necessary limitation of the interactive editor metering system. The total number of commands (TC) was a frequency count of the number of times commands were entered in the editing session, including during the 'fixing' session, if one was needed.

The ANOVAs performed on these dependent measures showed no significance for the editor by user interaction. Significance would have been expected if users had indeed per-

TABLE 6

Overall Dependent Measures of the Validation Study

TT	Includes all task time from beginning of actual editing to completion of edit task, including any 'checking' or 'fixing' necessary.
TE	A frequency count of all computer-rejected commands, whether typing mistakes or incorrect commands. Does not include acceptable commands which did not make the desired change.
TC	Total number of times a correct command was entered by the subject.
T	Time to complete editing one task (not including any 'fixing' time.)

formed 'better' on the appropriate model-based editor. There was also no significance for the factor of editor, which suggests no difference between editors over all users. In fact, significance was found only for the factors of user and practice, as shown in Table 7 . Not surprisingly, experts used less time and fewer commands than naives, and subjects took significantly less time to complete their second edit task (at $p<.01$). See Appendix O for the ANOVA summary table of this and subsequent analyses.

A comparison of cell to cell variance was also made to check the assumption of homogeneity of variance, since the experimenter had noticed wide variability between subjects. An F-Max text rejected the null hypothesis at $p<.01$ for all dependent variables, showing strong heterogeneity of variance.

It was hypothesized that these high cell to cell differences might be obscuring some actual differences in editors. Logarithm transforms were performed on the dependent measures in an attempt to reduce these variance differences. Heterogeneity was reduced for some measures by the log transforms, but the assumption of homogeneity was still rejected for all dependent variables at $p<.05$.

TABLE 7

Summary of Statistically Significant Effects:

Overall Dependent Variables

User Type				
	Novice		Expert	
DV	$\bar{X}$	s	$\bar{X}$	s
TT	24784.24	8469.33	14235.66	5865.06
TC	126.99	52.87	97.58	36.97

Practice				
	Session 1		Session 2	
DV	$\bar{X}$	s	$\bar{X}$	s
TT	21973.75	9030.17	217046.15	8284.36

It was noted that some subjects, at the end of a given session, had reviewed their edited file, checking to make sure all changes had been made. This 'checking' had the effect of increasing the TT, TC, and possibly TE values. Of the 212 files, 57 included some checking. The distribution across editors and across user (novice and expert) is shown in in Table 8 . A χ^2 test showed that the cells and different editors were not independent, but user was a significant predictor of checking at $p<.01$. In order to remove this possible cause of heterogeneity of variance, the 'checking' sequence of commands was removed from those files containing it. ANOVAs on TT, TC and TE were repeated on this reduced data set, but the results, both in terms of significant effects and continued heterogeneity of variance, agreed very closely with previous analyses.

The absence or presence of 'fixing' files was also examined as a possible source of variance. Forty-two of 80 subjects needed to 'fix' or take part in another session to perform missed edit changes. Twice as many of these 'fixers' were novices as experts. A breakdown by editor and user is shown in Table 9 . User was shown by a χ^2 test to be significant at $p<.01$. In order to remove this bias, T (time), a dependent variable which did not include the elapsed time of the 'fixing' session, was added to the ana-

TABLE 8

Checking Distribution

		EDITOR				
		Full	Novice Model	Control	Expert Model	TOTAL
USER	Novice	7	5	5	5	22
	Expert	8	10	10	7	35
	TOTAL	15	15	15	12	57/212 files

A χ^2 test showed user significant at $p<.01$.

lyses. It was reasoned that exclusion of the 'fixing' time would avoid significant inflation of the TT scores of some subjects since a large part of the 'fixing' time was due to searching, not correction of edit changes. This measure T showed significance only for user and practice, reinforcing earlier findings.

Other ways of reducing heterogeneity of variance were explored. Upon examination of the cell variances it was found that the full and control editors had double the variances of the editors based on the novice and expert user models. Analyses were performed only on the novice and expert model editor data, based on the assumption that the high variance was tied to the other two editors. Reducing the data to the model-based editors did not bring the variance differences down to a manageable level. The ANOVA results were also essentially the same as those of the initial analysis.

In another attempt to exclude the high variance data, analyses were performed on the expert subject data alone, since the novice subject variance was much higher than that of the experts. Once again, heterogeneity of variance was present. ANOVA results showed again only user and practice, not editor or the editor by user interaction as significant.

TABLE 9

"Fixing" Distribution

		EDITOR				
		Full	Novice Model	Control	Expert Model	TOTAL
USER	Novice	5	8	8	7	28
	Expert	2	3	3	6	14
	TOTAL	7	11	11	13	42/80 subjects

A χ^2 test showed independence of user group at $p<.01$.

Higher order interactions such as editor by practice and editor by text by practice, shown to be significant in several of these ANOVAs and in later analyses, will not be discussed. The same is true of the significant factor text and its interactions found in some analyses. Although the texts were intended to be equivalent, there are apparently differences which affect performance. Although these effects may be interesting in their own right, they do not help resolve the question of user model validity.

ANOVA of detailed dependent measures. Several fine-grained analyses were performed, exploring different dependent measures in an attempt to check for significant effects perhaps hidden by the high heterogeneity of variance of the overall dependent variables. These detailed dependent variables are listed in Table 10 .

The six basic commands (UP, DOWN, CHANGE, DELETE, AREA, and CHANGE TO INPUT MODE) were common to all four editors and included specifically in the introduction's fading session of editing. The ratio of the number of times these basic commands were used to the total number of times commands were used was calculated for each subject, forming the dependent variable B (basic). It was reasoned that novices would tend to have a higher ratio of basic to total commands

TABLE 10

Dependent Variables of the Detailed Analyses

B	The number of times the six basic commands (UP, DOWN, AREA, DELETE, CHANGE, CHANGE TO INPUT MODE) were entered divided by the total number of times commands were entered.
NC	Total number of times a novice command (i.e., a command contained in the novice model editor command set) was entered.
A	Frequency of use of commands which actually alter the file edited.
M	Frequency of use of commands which involve cursor movement.
A/M	The ratio of 'alter' commands to 'move' commands.

than experts, because they would rely more heavily on the six fundamental commands. This hypothesized result was significant ($p<.01$) even though the ANOVA had significant heterogeneity of variance ($p<.01$). In addition, editor was significant at $p<.01$. The means and variances of the four editors are shown in Table 11 . Newman-Keuls post-hoc comparisons, also included in Table 11, showed significant differences between some editor pairs. Subjects in the full editor had a significantly lower ratio of basic to total commands than subjects in the novice model and control editors. Users of the expert model editor had a lower ratio than users of the control editor. In fact, subjects using the control editor had a ratio of basic to total commands significantly higher than that of subjects in any other condition. The significance of the factor editor in the analysis of basic to total commands (B) implies that user strategy can be manipulated as a function of the editor command set. Selection of certain command sets could lead users to depend less on basic commands. This editor manipulation could lead to the learning and regular use of more powerful commands, and in this way introduce a more efficient editing strategy.

TABLE 11

Summary of Significant Editor Effect

in the Analysis of Basic Commands

	Editor							
	Full		Expert		Novice		Control	
DV	$\bar{X}$	s	$\bar{X}$	s	$\bar{X}$	s	$\bar{X}$	s
B	.7845	.1131	.8283	.1044	.8656	.0707	.9176	.0539

Results of Newman-Keuls Paired Comparisons, Editors

Full	Expert	Novice	Control

A further analysis looked at two complementary dependent measures : 1)the frequency of use of commands which actually altered the file (A), and 2)the frequency of use of commands which did not alter the file, generally involved in moving the cursor around within the file (M). The ratio of alter to move commands (A/M) was also included. These dependent measures showed significance only for user (M), text (A, A/M) and higher order interactions with text. There were no significant differences for editor or editor by user. Once again, heterogeneity of variance was present ($p<.01$)

The final supplemental analysis consisted of limiting the commands included in the analysis to those in the novice model editor, thus comparing the command set common to both model editors. The frequency of use of these commands became a new dependent variable (NC) which was compared in the novice model editor and the expert model editor conditions. Variance remained heterogeneous, and only the user main effect was significant ($p<.01$).

The results of the above described detailed analyses are summarized in Table 12 .

TABLE 12

Summary of Statistically Significant Effects:

Detailed Dependent Variables

	User Type			
	Novice		Expert	
DV	$\bar{X}$	s	$\bar{X}$	s
B	.8854	.0755	.8126	.1095
NC	119.03	45.14	93.78	25.22
M	71.40	35.87	56.31	30.08

Cluster Analysis. The second main analysis of the validation study data involved only those subjects who used the full editor, i.e., 20 subjects, because only these subjects had learned the same editor as all subjects in the model development study. (For the minor differences between the editors, refer to Table 5 .) However, although the editor and texts were the same both for the model development and the validation study subjects, there was the major difference of editing environment. The validation study subjects performed the editing interactively, not using paper and pencil, as did the model development study subjects.

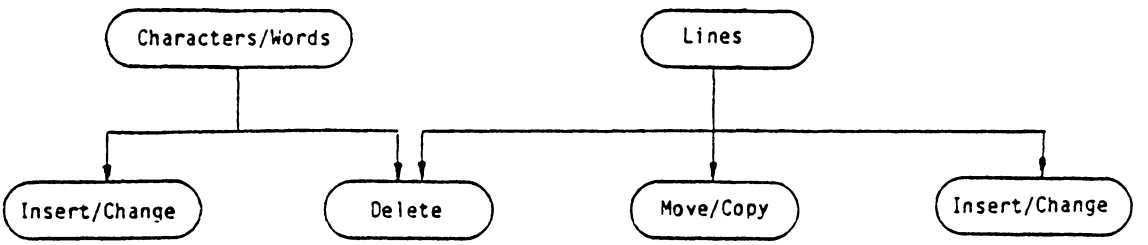
The cluster analysis of the validation study data is essentially a replication of the cluster analysis used on the model development study data. It was hypothesized that the cluster output from the actual interactive editing sessions would match or at least approximate the original cluster output. This would show that the 40 subjects who edited with paper and pencil had basically the same user models as the 20 subjects who edited interactively (using the equivalent full editor).

The first level of the cluster analysis consisted of grouping the edit changes (listed in Table 1) based on the commands used by the subjects to make them. This was done

for each individual subject's responses. There appeared to be less agreement than was found in the model development study. None of the subjects' cluster outputs were good approximations of the expected overall model. The original model, at this level, consisted of the four general clusters of inserting and changing characters and words; deleting characters, words and lines; moving and copying lines; and inserting and changing lines. In general, the cluster outputs of the validation study did not show these four edit change clusters.

The only similarities to the original model involved the grouping of line level edit changes. For two novice and five expert subjects, the move lines and copy lines edit changes were clustered together. The edit changes insert lines and change lines were also grouped together by one novice and three expert subjects. On a more general level, some cluster outputs showed an overall grouping of line-level edit changes which might include insert, change, delete, move and/or copy lines. Five experts showed this type of line-level cluster, sometimes including edit changes not on the line level, such as join and delete to string. Figure 3 shows an overall view of the expert subject cluster results as compared to the clustering of the model development study.

Model Development:



Validation:

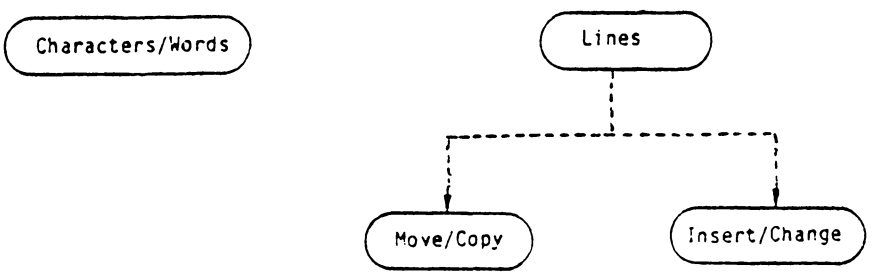


Figure 3: Expert Subjects, Model Development and Validation

All or part of these similarities to the original model were included in nine of the ten expert cluster outputs, and two of the ten novice cluster outputs. Edit changes other than those on the line level seemed not to be grouped systematically within each cluster output, or organized consistently across cluster outputs. Caution must be taken in interpreting these results as firm support of the models since these similarities were not only relatively minor, but also often did not occur at the level of four clusters. In other words, the output needed to be formed into more clusters (e.g., six or seven) in order to show these similarities. The clusters are not as clear cut as those of the model development study, even where similarities were found.

Within this allowance, one can state that there appeared to be more agreement among expert subjects than novices. As mentioned earlier, minor similarities could be found in nine of the ten expert cluster outputs, but only two of the ten novice cluster outputs. Not only did experts show more agreement with the original model, but also more agreement among themselves.

It might be hypothesized that this higher agreement among experts is due to their greater familiarity with interactive

already mastered certain basic concepts, they are less hampered by logistics problems. Their interactive editing behavior more closely matches their mental model of the editing task.

The variability between individual cluster outputs was large enough to prevent the second level of clustering analysis. Clustering of the commands used within each edit change cluster was not possible because the edit change clusters could not be clearly defined.

Frequency analysis. The difference between the model development and validation study cluster output might be due to the paper-pencil versus interactive editing environment. Since the cluster analysis is based on frequency counts, the differences in the cluster outputs may be isolated by analyzing the frequency counts of commands used. These frequency counts, listed in Table 13, point out the specific differences in command use between interactive and paper-pencil editing. Novice and expert subjects command frequencies are compared between the model development study and the full editor condition of the validation study.

The two frequency distributions were shown to be significantly different at $p<.01$ by a χ^2 test for independence. Since the frequency counts were measured across 20 subjects

TABLE 13

Command Frequencies: Development vs. Full Validation

Command	Dev.: Naives	Valid.: Full Ed., Naives	Dev.: Experts	Valid.: Full Ed., Experts
ADD	33	36	17	16
AREA	17	717	0	113
BOTTOM	7	13	8	15
CHANGE	65	238	146	301
CLEAR STORE	3	3	0	12
DELETE	149	222	97	118
DELETE UP	1	1	1	0
DELETE TO	14	4	18	0
DISPLAY	4	2	0	53
DISPLAY STORE	0	10	0	16
DOWN	362	498	348	600
FIND	51	4	40	46
FINDUP	6	0	13	7
INSERT	75	75	9	52
JOIN	11	15	24	19
MARK	8	1	12	5
PASTE	10	9	1	13
PASTE & ERASE	27	28	36	26
REPEAT	2	7	22	49
RETURN	8	0	11	5
SPLIT	15	43	24	34
CUT	15	8	17	20
CUT & ERASE	25	26	21	19
SWITCH	20	17	19	19
TOP	4	13	4	29
TRANSPOSE	19	10	6	18
UP	28	229	20	151
LINE	0	3	1	1
TO INPUT MODE	44	460	45	80
χ^2 statistic:	34926.629**		3869.462**	

** Significant at $p<.01$.

Since the frequency counts were measured across 20 subjects in the model development study conditions and 10 in the validation study conditions, the values of the model development study frequency counts were halved and used as the expected value for the χ^2 test. The χ^2 test results support the hypothesis that paper and pencil editing is different enough from interactive editing to explain the cluster output discrepancies. Looking at specific command frequencies, this shows up clearly. Two large areas of difference seem to be in commands which are involved in 1)finding position within the file, and 2)knowing whether input or edit is the active mode. For example, both novices and experts used the AREA command many times more often in the interactive environment. The AREA command simply displays a screen of text around the current line. Its much higher frequency of use in the interactive environment implies that subjects had trouble keeping track of their location in the text, a problem nonexistent in the paper-pencil editing environment. Similarly, DISPLAY (for experts), DOWN, UP and TOP had much higher frequencies for the interactive editing. These commands also are most likely to be used to find the current line or the relationship of the current line to the line to which one wishes to move. This may have resulted because a one dimensional line editor was used in the study. The fre-

quency difference would probably not have been present if a two dimensional editor had been used. In fact, the AREA command itself would not have been required.

Another large difference was present in the frequency of changes to input mode, especially for novices (44 with paper-pencil, 460 on interactive). Apparently the difference between input and edit modes was confusing to novice subjects, or they had trouble remembering which mode was active at a given time. It also seems apparent that novices, and to a certain extent experts editing interactively, made less use of some of the more powerful commands. Specifically, FINDUP, MARK, RETURN and DELETETO were used less in the interactive than in the paper-pencil editing. This indicates a tendency to stick to 'easier' commands with which the user is familar and a hesitancy to try the more powerful and complicated commands.

Especially in the case of novices, there seems to be a large discrepancy between the expected strategy and the strategy actually implemented. Novices expect to be able to perform the interactive editing following their paper-pencil strategy, but are not able to do so. The two largest discrepancies involve concepts alien to paper-pencil editing, namely movement of the current line (location within the

file) and changing between input and edit modes. These differences would be explained by the notion that certain concepts not involved in paper-pencil editing are simply not included in the naive user model. Perhaps any editor which requires the user to switch back and forth between input and edit, or is a line-oriented editor, can not truly match the user's model.

CONCLUSIONS

The results of the development and validation studies raise several questions concerning the development, validation, and implementation of user models of text editing. Although the data analyzed in the validation study did not produce evidence that novice subjects performed better on the novice model editor, and experts better on the expert model editor, several issues requiring additional research are readily apparent.

User Model Development

The results of the model development study showed high agreement among subjects, suggesting that the paper-pencil models developed do represent the users' internal model of interactive text editing. These models may not have been validated because the subjects' actual interactive editing behavior was limited by the constraints of the editor used. The user models supplied a model of the command set required, but perhaps this alone is not enough to fully describe the user's internal model.

Considerations of deeper structure may need to be included in the development of user models. Not only surface issues such as command set, but also deeper issues such as mode change and the need for 'location' commands (specifically, the AREA command) need to be addressed. In both of these cases, large differences in frequency of use between the paper-pencil and interactive editing environments indicate that they are aspects of interactive editing which must be considered.

Validation of User Models

The results of the validation study pointed out the possibility that certain aspects of interactive editing were not successfully modelled by the present approach. Perhaps the user models need to include not only a command set, but other specifications as well. These other considerations may have such a great effect that the different command sets of the four editors had very little impact in comparison to these larger issues. In other words, the fact that none of the editors was modelled in terms of these deeper considerations so outweighed any differences in command set that performance was essentially the same across all editors. In effect, the model editors may have reflected the user models, but incompletely.

Valid models. There are other possible reasons explaining why the correct models would not have shown up performance differences. It is possible that the high heterogeneity of variance was obscuring significant effects. This high subject variability seems to be a characteristic of attempts to model interactive editing behavior (for example, Card, Moran, and Newell, 1980, and Embley et al., 1978). In the present validation study, the overall and fine grained analyses were for the most part unsuccessful in eliminating this heterogeneity of variance. Nevertheless, the ANOVA test is very robust, and would most likely still be valid.

Perhaps the procedure used to evaluate the models was inappropriate. There may have been better dependent measures, such as the number of errors in terms of attempted but unsuccessful command entries, or time spent actually editing vs. time accounted for by mental activities.

The fault may lie with the validation method itself. There may have been a better way to determine the user model. For example, performance differences due to editor may only appear under high workload or stress. Another source of difference might be the experimental situation. The short term editing in the validation study may not accurately replicate the typical editing situation, and produce

different results. Long term monitoring of system users' command frequency counts would eliminate the contrived experimental situation, perhaps showing model validation.

Invalid models. On the other hand, the ANOVA results may be correct, accurately showing no empirical differences between the four editors tested in this study. It may be the case that in the interactive environment, users have so many different strategies that overall models are simply not appropriate. These varying strategies may be due to differences in ability to use certain commands successfully, differences in willingness to try new commands, or differences in systems used in the past.

If overall models are appropriate, the particular models developed may be incorrect. The paper-pencil format may have failed to simulate sufficiently the actual interactive environment and so gave erroneous predictions of interactive user command set models. Another possibility is that the type of clustering algorithm used simply was not an appropriate method for forming the user models. Additionally, the criteria used in determining the number of clusters, or the subjective judgement used to come up with these criteria may have been faulty.

The explanation for the lack of validation may be more basic, though. Actually working with an interactive editor differs from doing editing by hand. In using the interactive system, there may be a relationship between the learning of commands and editing strategy. In the paper-pencil environment, the subject need only believe he or she understands the command and its function to use it successfully, but in the interactive environment, problems of the command's actual use may arise. The subject must then use a different command or commands to accomplish the change, and then may adhere to that usable strategy in the future. The problems would arise not only in the basic function of the command, but also in the details of its use--such as specifying the line on which the current line pointer must be and whether input or edit mode should be active. The user may have a model of the editing task in his or her head, but when limited within the interactive editor's framework, cannot implement the model.

This basic difference between paper-pencil editing and interactive editing could be the explanation for a difference in commands used to make a given edit change. A subject might enter one series of commands to make a given edit change if working with paper and pencil and another if working in an actual interactive environment. In fact, this

study shows just such a difference not only in the ^{2} tests on frequencies of commands across the model development and validation studies, but also in the results of the cluster analyses performed. There was very little agreement among subjects in the interactive validation study as contrasted with the high agreement present in the model development study cluster output. Within the validation study, the higher agreement among expert subjects would suggest that as subjects become more experienced, they overcome difficulties in using certain commands which led to avoidance of those commands in the first place. It seems that experts are able to more closely approximate their internal user's model when editing interactively than are novices. Because of this, expert editing behavior shows more agreement across subjects.

A further explanation might be that interactive editing is so different from paper-pencil editing that the interactive system is simply not compatible with the user's model, especially in the case of the naive user. This also is supported by the lower agreement in cluster outputs among naive subjects as compared to expert subjects in the validation study.

Research Implications

The most obvious of the issues raised by this research is the need to include deeper aspects of interactive editing in user models. The large descrepancy in frequency counts between paper-pencil and interactive editing in the areas of mode change and current line location indicate that these are two deeper aspects which deserve attention. Research evaluating the importance of these issues and investigating other such features is needed.

Related to this is the need for the comparison of one and two dimensional editors. One dimensional editors, in which the cursor may be moved anywhere on the screen (not just within the current line) would eliminate the problem of current line location.

Further investigation of one dimensional editors, however, in which modelling, as well as validation, was done in an interactive environment could prove fruitful. The results could show whether the original paper-pencil format and cluster analysis, or the lack of inclusion of deep aspects (e.g., mode change and line location) were responsible for the lack of validation. However, the high agreement in user models yielded by the paper-pencil format and cluster analysis suggest that these models are appropriate.

Considering the descrepancies in paper-pencil and interactive editing frequencies of commands related to deeper issues, the exclusion of these issues from the models seems more likely to be the cause of lack of validation.

How can these deeper aspects of editing be included in user modelling? The user models describe the user's world view. Perhaps we already have that view in the paper-pencil editing format. We may need to make interactive editors as similar to paper-pencil editing as possible (but faster). The electronic version would then be compatible with the user model fashioned after paper-pencil editing.

The two deeper issues particularly noticeable in the present research were mode change and current line location. Mode change is not present in paper-pencil editing and is a concept to which new users must adjust. If it were eliminated, interactive editing would more closely match the user model. By having the editor automatically switch between modes, the user would not have to make that adjustment. Such automatic switching could be done through the use of an editing keypad. Each command would be represented on a key. Any use of the actual keyboard would be correctly interpreted by the editor as being new text input.

Confusion between modes could also be reduced by clear indication of the presently active mode, either by text arrangement on the screen or auditory or visual signals. Using voice or touch input for mode change might also reduce confusion.

The problem of current line location would be eliminated through the use of a two dimensional editor, as mentioned above. In using a one dimensional editor, a display of the area around the marked current line after each command entry might also help.

Models including not only command set but also consideration of deeper issues would have important implications in model development. Editors might be tailored to the specific user's experience level. Novices would start by learning the novice model command set, and would learn additional commands as they gained experience. These additional commands, leading to the expert model command set and finally the full command set, could be introduced either by the user or the computer, based on hours of use, number of times less efficient commands were used, or other criteria.

The results of the present model development and validation studies indicate that user models can be developed by use of the clustering algorithm. The between-editor differ-

ences in the ratio of basic to total commands suggest that manipulation of the command set can affect user strategy. However, user models of interactive editing must consider not only command set, but also deeper issues such as mode change and current line location. Modelling of these deeper aspects of editing needs to be explored, as does the implication of two dimensional vs. one dimensional editor use and the possibility of development of new editors which more closely match the user model at all levels.

Appendix A

MODEL DEVELOPMENT STUDY INTRODUCTION AND COMMAND DESCRIPTIONS

What you'll actually be doing for this experiment is fairly simple--you'll be making editing corrections to a manuscript. This involves taking out parts of text, changing other parts, putting in new text, etc.

You'll be working with pencil and paper, but are going to pretend you're on an interactive computer terminal with a keyboard and screen and using the text editor which is built into the system. In order to do this you need to know more about interactive terminals and editors.

THE BASIC CONCEPTS OF INTERACTIVE TEXT EDITING

Interactive users communicate with the editing system through a terminal. A terminal looks like and sometimes behaves like a typewriter, except that it is connected to a computer. You tell the editing system, or Editor, what you want it to do by typing commands at the terminal. The editor responds by performing your request, and both your typed input and the editor response are displayed on the terminal screen.

When editing a manuscript, there are two modes. The first is called INPUT mode. When the editing system is in INPUT mode, anything that you type on the keyboard will appear on the screen exactly as you type it in, and it is stored on the electronic sheet of paper. This is much like the way that you start with a blank piece of paper in a typewriter and gradually fill it.

The second mode is called EDIT mode. In this mode, what you type on the keyboard appears on the screen, but the editor interprets it as an editing command.

You enter INPUT mode by using the command INPUT, and typing the new text on the next lines. When you have finished typing in new text, you can move into the EDIT mode by typing the EDIT command. (This command won't be read as new text input.)

When you start editing, the editor will be in EDIT mode until you give it the INPUT command. Make sure you're in the right mode for what you want to do.

Before you start the actual editing, you'll become familar with some editing commands. These are the words or phrases which you would type in at the terminal's keyboard, "commanding" the editor to do a specific operation. Instead of retyping or rewriting the manuscript by hand, you choose the commands you would type on the terminal keyboard to get the editor to perform the desired operation.

Even though you'll be working with pencil and paper, you need to visualize the text as though it is actually appearing on a terminal screen. In order to do this you need to know how the terminal displays the text. Basically, the editor stores text as though it's on one long sheet of paper, but with only part of it displayed at a time. In the editor you'll be working with, the "window", or the part of the text displayed at one time, is only a single line; the current line. (Certain commands can be used to temporarily display more lines of text). In this case, the editor uses a LINE POINTER to indicate the current line, usually the last line displayed. Most edit commands will only make changes on the current line.

Remember that even though you're actually working with paper and pencil you need to make sure you have moved the imaginary line pointer to the right line before making a change. Also, remember that the current line is the only one you'd be able to see on the actual terminal screen.

COMMAND NAMES

Commands	Arguments
ADD	*
AREA	n CHARS
BOTTOM	n LINES
CANCEL	n WORDS
CHANGE<string1/string2 >	RTL
CLEAR STORE <n, * >	
COPY <n LINES, n WORDS >	
DELETE <string, n LINES, * >	
DELETE UP <n, * >	
DELETE TO < string >	
DISPLAY< n, * >	
DISPLAY STORE <n, * >	
DOWN <n >	
EDIT	
FIND < string >	
FINDUP < string >	
INPUT	
INSERT	
JOIN	
LEFT <n>	
MARK	
<n>	
PASTE <n>	
PASTE AND CLEAR <n>	
REPEAT	
RETURN	
RIGHT < n >	
SPLIT < string >	
STORE < n>	
STORE AND ERASE < n>	
SWITCH<word1/word2 >	
TOP	
TRANSPOSE	
UP<n >	
#	

COMMANDS

Following is the list of editor commands you'll be using, including names and descriptions. In the command names, the abbreviations which may be used are in underlined capital letters. In listing the commands, you may use either the full name or the abbreviation. In the descriptions, these symbols will be used:

>	This prompt marks the user input. Anything next to a prompt like this was typed by the user (i.e. you). All other lines are displayed by the editor (i.e. the computer).
LP →	The line pointer indicates which line is the current line. It moves only vertically. At the start of editing, its position is above the first line of the manuscript.
"n"	Indicates a changeable number.
"string"	Refers to a specific phrase or series of characters within the text.
< >	Within these brackets are the arguments which can be used with the command listed. (The brackets don't need to be included in the typed command.)
+	The cursor indicates a location within a given line and moves only horizontally within the current line. At the start of editing, the cursor is to the left of the first character in the line and returns to the far left every time you start a new line.

ADD <string>	Adds text to the end of an existing line (i.e. appends a string onto a line). Full line is displayed on the terminal screen. Total line must not exceed page width.
AREA	Displays three lines above and three below the current line. When the next command is issued, once again only the current line is displayed.
Bottom	Moves the line pointer to the line below the last line of text, i.e. to the bottom of the manuscript.
CANcel	Reverses modifications made by the most recent successful edit command; as though it had not been issued. The corrected text is displayed.
Change string1/string2 (may be used: *)	Replaces unwanted text with wanted text by changing string1 to string2 within the current line. If '*' is specified, string1 is changed to string2 from the current line to the bottom of the manuscript (including the current line).
Clear Store <n, *> (n=number of lines)	Erases a specified number of lines which have been stored in the 'waiting area' (starting with the first line stored). If '*' is specified, all stored text is erased.
COPY <n lines, n words>	Duplicates a set of text, resulting in two copies of that text portion. If words are being copied, the cursor must be placed at the first letter of the first word to be be copied. When copying lines, the line pointer location determines the first line copied (i.e. the current line). The new lines are displayed on the terminal screen.
DElete <string, n lines, *>	Removes segments of text from the original manuscript. If a string is specified, all ocurrences of the string within the current line are deleted. If n lines are specified, the current line is the first line deleted. If '*' is specified, all text to the bottom of the the manuscript, starting with the current line, is deleted.
DELete UP <n, *> (n=number of lines)	Deletes n lines, starting with the current line, up from the current line. If '*' is specified, all text, to the top of the manuscript, starting with the current line, is deleted. The line pointer will then be above the first line of text.

DELete TO ⟨string⟩	Erases the text up to a certain word or phrase, starting at the beginning of the current line.
DISplay ⟨n, *⟩ (n=number of lines)	Displays a specified number of lines on the terminal screen, starting with the current line. If '*' is specified, all text from the current line to the bottom of the manuscript is displayed. If no number of lines is specified, the current line is displayed.
Display Store ⟨n, *⟩ (n=number of lines)	Displays on the terminal screen the specified number of stored lines (i.e. those in the 'waiting area') starting with the first line stored. If '*' is specified, all lines in the 'waiting area' are displayed.
Down ⟨n⟩ (n=number of lines)	Moves the line pointer downward from the present position, and displays the new current line.
Find ⟨string⟩	The editor searches for a particular string of text from the current line to the bottom of the manuscript, moves the line pointer down to that line, and displays the line on the screen. If the string is not found, the line pointer remains at the bottom of the manuscript.
FindUP ⟨string⟩	This is the same as the FIND command, but searches the text from the current line up to the top of the manuscript.
INPUT	Readies the editor to accept new lines of text which the user will type in. The EDIT command must be used to return to the editing mode.
Insert	Readies the editor to accept a string (words or characters) to be inserted somewhere in a line. The cursor should point to the character in front of which the string will be inserted.
Join	Joins the current line and the next line into one, which replaces the current line. The new line must not be wider than the page, and will be displayed on the screen.
Left ⟨n⟩ (n=number of characters)	Moves cursor left within the current line of text.
Mark	Saves the location of the current line so it can be returned to later after editing changes have been made elsewhere in the manuscript.
⟨n⟩	Moves the line pointer to the line of the specified number.

Paste ⟨n⟩ (n=number of lines)	Moves a specified number of lines from the 'waiting area' into the text after the current line.
Paste and Clear ⟨n⟩ (n=number of lines)	Moves the specified numver of lines from the 'waiting area' into the text after the current line and clears these lines from the 'waiting area'.
REPeat	Repeats the most recent successful command.
RETurn	Returns to the line specified by the MARK command.
Right ⟨n⟩ (n=number of characters)	Moves the cursor right from the present position within the current line.
SPlit ⟨string⟩	Breaks the current line into two lines before the specified string. Both lines will be displayed.
Store ⟨n⟩ (n=number of lines)	Places a specified number of lines, starting with the current line, inot a 'waiting area', to be used later.
Store and Erase ⟨n⟩ (n=number of lines)	Places a specified number of lines, starting with the current line, inot a 'waiting area' to be inserted or used later, and erases them from the text manuscript itself.
SWitch word1/word2	Exchanges the position of two words. The operations will be performed on the first found occurrence of each word1 and word2 within the current line.
Top	Moves the line pointer to the line above the first line of text, i.e. to the top of the manuscript.
TRanspose char1/char2	Transposes the positions of two consecutive characters. The cursor should be pointing to the left member of the character pair.
Up ⟨n⟩ (n=number of lines)	Moves the cursor upward the specified number of lines from the present position and displays the new current line on the screen.
	The editor responds to this symbol by displaying the number of the current line.

Arguments

Arguments may be used with commands. They give the editor more details about the desired operation. If an argument is used, it is typed to the right of the command name. The following are arguments you may use:

*	This means "all". For example, operation is to be performed on every occurrance of the specified pattern.
n CHars	Specifies the number of characters on which an operation is to be performed.
n Lines	Specifies the number of lines on which an operation is to be performed.
n Words	Specifies the number of words on which an operation is to be performed.
RTL	Makes edit changes from right to left within the current line. For example, in a search, the matching string of text furthest to the right would be acted on first.

Appendix B

INFORMED CONSENT FORM

42
COLLEGE OF ENGINEERING

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

Blacksburg, Virginia 24061

HUMAN FACTORS LABORATORY, DEPARTMENT OF INDUSTRIAL ENGINEERING AND OPERATIONS RESEARCH (703) 961-5358

PARTICIPANT'S INFORMED CONSENT

As a participant in this experiment, you have certain rights. The purpose of this note is to make you aware of these rights and to obtain your consent to participate.

1. You have the right to stop the experiment in which you are participating at any time if you feel that it is not agreeable to you. Should you terminate the experiment, you will receive pay only for the proportion of time you participated.

2. You have the right to see your data and to withdraw it from the experiment if you feel that you should.

3. You have the right to be informed on the results of the overall experiment. If you wish to receive information on the results, please include your address (three months hence) with your signature below. A summary will be sent to you. If you would then like further information, please contact the Human Factors Laboratory and a full report will be made available to you.

We hope you will find the experiment a pleasant and interesting experience. The faculty and graduate student involved greatly appreciate your help as a participant. If you have any questions about the experiment or your rights as a participant, please do not hesitate to ask. We will do our best to answer them, subject only to the constraint that we do not want to pre-bias the experimental results.

Your signature below indicates that you have read your above stated rights as a participant and that you consent to participation. If you include your printed name and address below, a summary of experimental results will be sent to you.

_____

SIGNATURE

_____

_____

_____

Printed name and address if you wish to receive a summary of experimental results.

Appendix C

MODEL DEVELOPMENT STUDY EXPERT EXPERIENCE PROFILE

---

1) How many school courses have you taken which involved a significant amount of interactive terminal use?

Median: 6 courses

Range: 0 to 15 courses

2) On the average, about how many hours per week have you been working at interactive computer terminals?

Median: 25 hours

Range: 4 to 60 hours

3) How long (in weeks, months or years) have you been working at interactive computer terminals?

Median: 3 yrs.

Range: 2 1/2 mos. to 6 yrs.

4) How long has it been since you last used an interactive terminal?

Median: 3 hrs.

Range: now to 3 wks.

5) Try to estimate the total number of hours you've spent

interactively on the computer.

Median: 1084 hrs.

Range: 45 to 16800 hrs.

6) How would you rate your skill in using interactive editors on a scale from one (low skill)to ten (high skill)?

Median: 8

Range: 7 to 10

Appendix D

BEFORE 0 TEXT

There are many kinds and degrees of growth; the type that I am going to focus on is the incremental growth experienced by small companies, of 200 employees or less [CURRS79], as a result of steadily expanding sales. The problem is particularly acute for small businesses because of the scarcity of human and equipment resources.

In large corporations, it is not uncommon to have fifty or one hundred people in a department, e.g. accounting, with carefully delineated responsibilities for it's staff, and a clear line of authority. In a small company, the accounting department may be comprised of just two or three people, with multiple responsibilities, and supervision may be overlapped by several people, not all of them accounting personnel. In addition to standard accounting tasks such as accounts payable and accounts receivable, personnel in the accounting department may also be responsible for handling cash sales, inventory control, answering and forwarding telephone calls, and other general secretarial duties.

Other departments in a small company will be organized along similar lines. The sales or marketing department may consist of the president or vice-president of the firm supervising one or two salesmen, or in a slightly larger company, there may be a sales manager who spends much of his time doing routine sales work. Because these personnel have to split their time, there may be problems
The production management team may again just consist of one of the officers of the company and one or two foremen. It would be unusual to find full-time white-collar plant

① Show a screen of text by entering:

AREA <return>

② Change this word to "companies" by entering:

DOWN 4 <return>
DELETE 1 <return>
UP 1 <return>

Go into INPUT mode by hitting <return>
Re-type the entire line using 'companies' instead of 'businesses'
Go back to EDIT mode by hitting <return> <return>

③ Remove the quote marks by entering:

DOWN 11 <return>
DELETE 1 <return>
UP 1 <return>

go into INPUT mode and re-type the line correctly, then return to EDIT mode.

④ Add this sentence by entering

DOWN 8 <return>

Then delete the line, back up one line to set the line pointer, and go into INPUT mode to enter the two lines.

managers, production schedulers, purchasing agents, and other specialized support personnel common to larger organizations.

In summary, the hierarchy tends to be both shallow and narrow. There may only be one or two levels of authority between the lowest and the highest person in the company, and those in authority are probably supervising more than one functional area in the company [CURRS79].

It is against this backdrop that a steady rate of grwoth represents such an insidious problem for the small business. In a much larger company, a growth rate of ten percent of gross sales per year is much easier to handle. For example, if a ten percent grwoth rate increases the workload ten percent in an accounting department of one hundred people, few problems would be encountered in increasing the staff by ten percent (ten people). The personnel department is notified, which takes the responsibility for advertising the position, screening applications and conducting interviews, and selecting the new employees. Once hired, one of the seven or eight supervisors in the accounting department conducts an intensive training program for a week in the company classroom, carefully laying out the duties and responsibilities of the new workers in the company.

(5) change 'larger' to 'large' by entering
DOWN 6
CHANGE /larger/large/

(6) Using 'DOWN' and any other commands, correct the spelling of these two lines.

(7) Change to "employed".

(8) add to the end of the sentence

when you have finished, type
STOP <return>

Appendix E

BEFORE 1 TEXT

B(

Statement of the Problem

Unquestionably one of the very prodigious machines of the twentieth century is the computer. Modern technology makes it ~~possible~~ feasible to consider digital computer applications to most tasks performed by people.

Clearly, both the behavioral scientist and the computer ~~lkhfhjvfmlk j jkeje;ljfi;knokhnigujab kjy hg hguklsdjloe ks=ijq hkhuyrdkhg lhig kjg kgjg kggljfdk kjh;o9 fvydh~~ scientist face the exciting research challenge of optimizing the use of computers in order to improve human operator productivity.

The design and concept of a computer demands a direct interface between the machine and the processing information capabilities and limitations of ~~the~~ people.

Computers are so commonplace in our modern society that it is difficult to imagine a situation in which computer technology does not impact one's daily life. Recent advances in this high-level digital computer technology suggest that the proliferation of large main-frame computers, minicomputers and microprocessors will only continue.

Licklider (1960) provided a provocative prognosis for future interactions between humans and computers. Rather than consider the computer merely as a mechanically extended person, he posed a symbiosis of the two. Such a human-computer relationship is characterized by a close cooperative partnership whereby each exists to

the other's mutual benefit.
~~the computer merely as a mechanically extended person.~~
The human and the computer combine to process data and
~~University faculty who have an extensive research history~~
formulate solutions in innovative ways. ~~in a variety of~~

Much of the research during the 1960s and 1970s was devoted to an improvement in human-computer interactions. Many research advances have been made in this area, but even today designers have extreme difficulty in using these heterogeneous research findings to specify optimal information processing interfaces between the human operator and the computer.

Possible Solutions // Solution #1 and Solution #2

~~specific display formatting techniques to the general design principles; the diversity of such principles is striking. Clearly, both behaviorists and scientist will~~

<<<<Please type STOP when you are finished.>>>>

Appendix F

AFTER 1 TEXT

Statement of the Problem

Unquestionably one of the very prodigious machines of the twentieth century is the computer. Modern technology makes it feasible to consider digital computer applications to most tasks performed by people.

Computers are so commonplace in our modern society that it is difficult to imagine a situation in which computer technology does not impact one's daily life. Recent advances in this high-level digital computer technology suggest that the proliferation of large main-frame computers, minicomputers and microprocessors will only continue.

Clearly, both the behavioral scientist and the computer scientist face the exciting research challenge of optimizing the use of computers in order to improve human operator productivity. The design and concept of a computer demands a direct interface between the machine and the information processing capabilities and limitations of people.

Licklider (1960) provided a provocative prognosis

for future interactions between humans and computers. Rather than consider the computer merely as a mechanically extended person, he posed a symbiosis of the two. Such a human-computer relationship is characterized by a close cooperative partnership whereby each exists to the other's mutual benefit.

The human and the computer combine to process data and formulate solutions in innovative ways.

Much of the research during the 1960s and 1970s was devoted to an improvement in human-computer interactions. Many research advances have been made in this area, but even today designers have extreme difficulty in using these heterogeneous research findings to specify optimal information processing interfaces between the human operator and the computer.

Possible Solutions

Solution #1 and Solution #2

Appendix G

BEFORE 2 TEXT

B2

When one attempts to generalize across the variety of specific display formatting techniques to more general design principles, the diversity of such principles is striking. For example, Martin (1973) recommended a set of eleven guidelines to be used in formatting interactive displays, whereas Shneiderman (1979) summarized some design goals put forth by various individuals for the more general human-computer dialogue considerations. A representative summary of proposed design guides is given in Table 2.

Table 2. List of Principles for Interactive Display Dialogues

Hansen (1971)

1. Know the user (i.e., faults and strengths)
2. Minimize memorization
3. Optimize operations
4. Engineer for errors

Many of these principles/goals are so general that they become intuitively obvious, whereas others suggest areas of needed research documentation. Principles of this nature are of little use to the analyst unless data are provided.

Miller (1973)

1. Present small amount of information at a time

B2

2. List one idea per display // 3. Clean up screen

4. Adapt to user's ability

5. Insure that computer responds always

6. Design formats for clarity

7. Strive for ~~alikeness~~ similarity

8. Avoid redundancy in dialogue

9. Avoid difficult words and characters

10. Allow user to correct errors

11. Provide for ~~very~~ easy means of correction

12. Give ~~very~~ prominent instructions

13. Keep the user informed

14. Provide readily acce/ssable/ he/lp

Wasserman (1973)

1. Provide for every possible user input

2. Minimize required user knowledge of computer

3. Provide explicit diagnostics and user assistance

Pew and Rollins (1975)

1. Provide for every possible user input

2. Respond consistently and clearly

3. Capitalize on user's knowledge basis

4. Adapt wordiness to the user's needs

Kennedy (1974)

1. Use terse "natural" language

2. Use short entries

3. Maintain "social element" to the communication

4. Permit user to control length of cues and errors

Appendix H

AFTER 2 TEXT

When one attempts to generalize across the variety of specific display formatting techniques to more general design principles, the diversity of such principles is striking. For example, Martin (1973) recommended a set of eleven guidelines to be used in formatting interactive displays, whereas Shneiderman (1979) summarized some design goals put forth by
various individuals for the more general human-computer dialogue considerations. A representative summary of proposed design guides is given in Table 2.
Many of these principles/goals are so general that they become intuitively obvious, whereas others suggest areas of needed research documentation. Principles of this nature are of little use to the analyst unless data are provided.

Table 2. List of Principles for Interactive Display Dialogues

Hansen (1971)

1. Know the user (i.e. Know the user faults and strengths)
2. Minimize memorization

3. Optimize operations

4. Engineer for errors

Miller (1973)

1. Present small amount of information at a time

2. List one idea per display

3. Clean up the screen

4. Adapt to user's ability

5. Insure that computer always responds

6. Design formats for clarity

7. Strive for similarity

8. Avoid redundancy in dialogue

9. Avoid difficult words and characters

10. Allow user to correct errors

11. Provide for easy means of correction

12. Give prominent instructions

13. Keep the user informed

14. Provide readily accessable help

Wasserman (1973)

1. Provide for every possible user input

2. Minimize required user knowledge of computer

3. Provide explicit diagnostics and user assistance

Pew and Rollins (1975)

1. Provide for every possible user input

2. Respond consistently and clearly

3. Capitalize on user's knowledge basis

4. Adapt wordiness to the user's needs

Appendix I

BEFORE 3 TEXT

B3

Overall Analysis.

Proliferation of computers in most Navy systems requires careful consideration of the human-computer interface in order to optimize human performance. // Specifically, the management of human-computer dialogues is needed to enhance the information processing and decision making capabilities of individuals working in real-time, demanding environments.

Overall Analysis.

The objective of this three-year proposed effort is to optimize human-computer communications.

A multidisciplinary faculty research team of behavioral and computer scientists will conduct programmatic ~~basic~~ research on six interrelated tasks which center around a ~~basic~~ system called a Dialogue Management System that is used to establish the human-computer interface. ~~DMS can be used to manipulate and change the human-computer interface.~~ DMS provides an efficient device-independent means for tailoring the human-computer interface. It will be used to investigate the human factors issues of engineering usable software, interactive display principles, voice input/output ~~decision making tasks for people working in various areas~~ interfaces, decision aiding and task ~~division~~ allocation. Existing research facilities in both the Human Factors Laboratory and the Computer Science Department will be integrated for the conduct of the extensive research. University faculty who have proposed research background in a variety of ~~human-computer, human-performance, computer science, and~~ human-computer, human-performance, computer science, and information science problems will work together as an

B3

interdisciplinary team on each of these related tasks. Our goal is to establish a center of excellence or interdisciplinary basic research on human-computer interactions.

To supplement available faculty expertise, an industrial advisor and a Joint Navy Scientific Advisory Committee will also be used. Close Navy coordination will be maintained throughout the research program in order to facilitate the transition of our basic research results to operational Navy problems. In this way, the end goal of the program will be realized.

Appendix J

AFTER 3 TEXT

Overall Analysis.

Proliferation of computers in most Navy systems requires careful consideration of the human-computer interface in order to optimize human performance.

Specifically,

the management of human-computer dialogues is needed to enhance the information processing and decision making capabilities of individuals working in real-time, demanding enviroments.

Overall Analysis

The objective of this three-year proposed effort is to optimize human-computer communications. A multidisciplinary facult research team of behavioral and computer scientists will conduct programmatic research on six interrelated tasks which center around a system called a Dialogue Management System that is used to establish the human-computer interface.

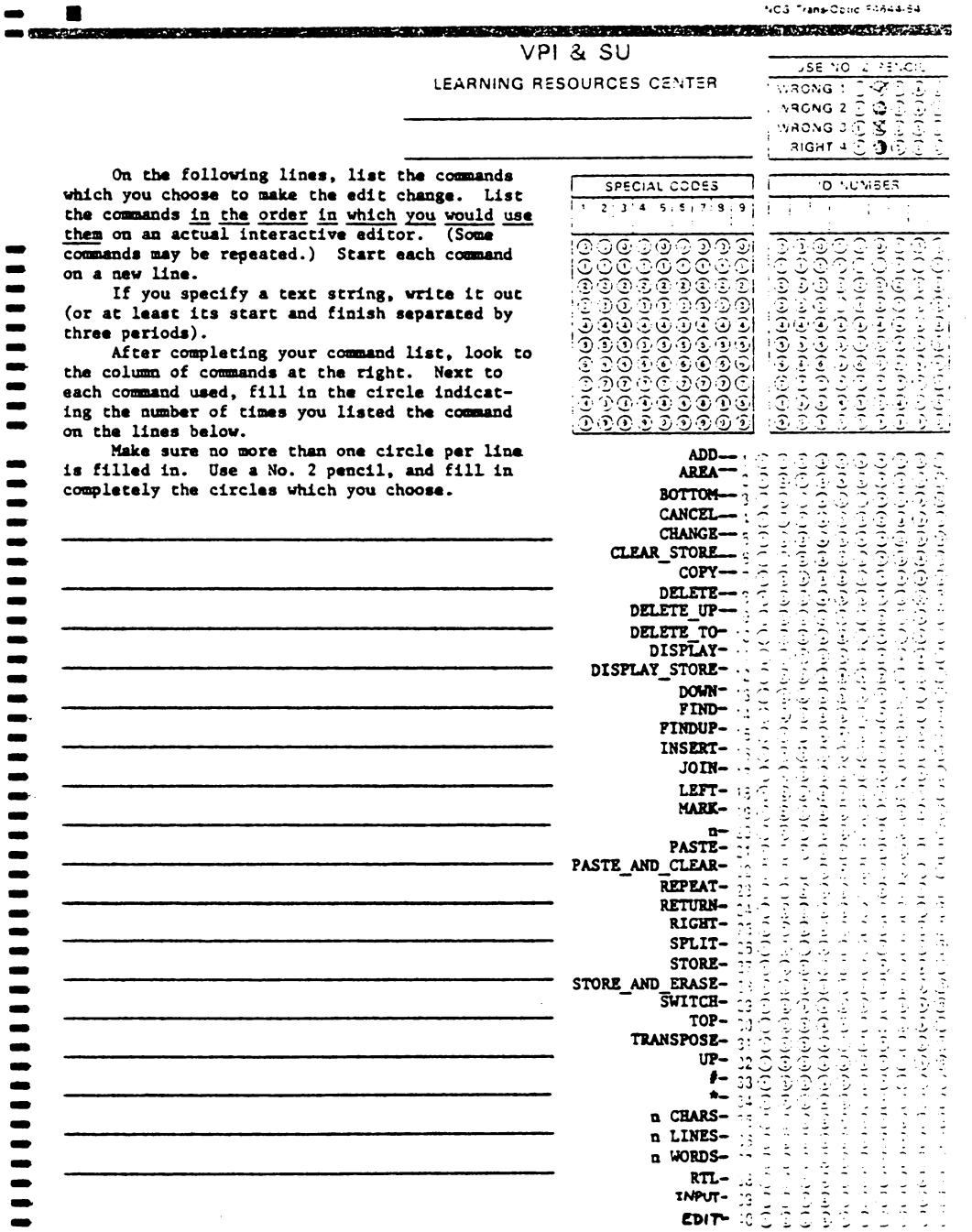
DMS provides an efficient device-independent means for tailoring the human-computer interface. It will be used to investigate the human factors issues of engineering

usable software, interactive display principles, voice input/output interfaces, decision aiding and task allocation. Existing research facilities in both the Human Factors Laboratory and the Computer Science Department will be integrated for the conduct of the extensive research. University faculty who proposed have research background in a variety of human-computer, human-performance, computer science, and information science problems will work together as an interdisciplinary team on each of these related tasks. To supplement available faculty expertise, an industrial advisor and a Joint Navy Scientific Advisory Committee will also be used. Close close Navy coordination will be maintained throughout the research program in order to facilitate the transition of our basic research results to operational Navy problems. In this way, the end goal of the program will be realized.

Our goal is to establish a center of excellence or interdisciplinary basic research on human-computer interactions.

Appendix K

MODEL DEVELOPMENT STUDY SAMPLE ANSWER SHEET



VPI & SU

LEARNING RESOURCES CENTER

USE NO. 2 PENCIL

WRONG 1

WRONG 2

WRONG 3

RIGHT 4

On the following lines, list the commands which you choose to make the edit change. List the commands in the order in which you would use them on an actual interactive editor. (Some commands may be repeated.) Start each command on a new line.

If you specify a text string, write it out (or at least its start and finish separated by three periods).

After completing your command list, look to the column of commands at the right. Next to each command used, fill in the circle indicating the number of times you listed the command on the lines below.

Make sure no more than one circle per line is filled in. Use a No. 2 pencil, and fill in completely the circles which you choose.

SPECIAL CODES

ID NUMBER

ADD—
AREA—
BOTTOM—
CANCEL—
CHANGE—
CLEAR_STORE—
COPY—
DELETE—
DELETE_UP—
DELETE_TO-
DISPLAY-
DISPLAY_STORE-
DOWN-
FIND-
FINDUP-
INSERT-
JOIN-
LEFT-
MARK-
n-
PASTE-
PASTE_AND_CLEAR-
REPEAT-
RETURN-
RIGHT-
SPLIT-
STORE-
STORE_AND_ERASE-
SWITCH-
TOP-
TRANSPOSE-
UP-
#-
*-
n CHARS-
n LINES-
n WORDS-
RTL-
INPUT-
EDIT-

Appendix L

EXPLANATION OF CLUSTERING ALGORITHM

The clustering algorithm used in this research derives a distance matrix based on frequency counts. The equation which yields these distance values is:

$$D(u, v) = \sqrt{\sum_{i=1}^{n}(x_{iu} - x_{iv})^2},$$

where u is the u^{th} item,

v is the v^{th} item ($u \neq v$),

i is the number of categories across which items u and v appear,

x_{iu} is the number of times the u^{th} item appeared in the i^{th} category,

and x_{iv} is the number of times the v^{th} item appeared in the i^{th} category.

Cluster analyses were performed at two levels in the present research. First, edit changes were clustered, based on the commands used to make them. In comparing two edit changes (u and v), x_{iu} would be the number of times command i was used to make edit change u. For example, in comparing the

two edit changes delete word and insert character, computation of the distance values would be done as follows. The number of times command 1 was used to insert a character (x_{1v}) would be subtracted from the number of times command 1 was used to delete a word (x_{1u}) and this quantity squared. The same would be done for command 2, command 3, and so on up to command n. These quantities would be summed, and the square root taken. This value would be the distance, in Euclidean space, between the edit changes delete word and insert character. This calculation is performed for all n(n-1)/2 or 190 edit change pairs, forming a distance matrix.

At this point, the grouping of edit changes may begin. At the level of weakest clustering each edit change is an individual cluster. The two clusters separated by the shortest distance are combined to form one. This is repeated, reducing the number of clusters by one each time a new joining is made. Eventually, all edit changes are included in a single cluster.

The two clusters to be joined together at a particular stage of the cluster analysis are determined in the following way. First, the diameter of each cluster is determined. For single point clusters, the diameter is defined to be

zero. For clusters consisting of multiple points, line segments are drawn between each pair of points within the cluster. The longest of these segments is that cluster's diameter. At a given step in the clustering process, the maximum diameter is defined as the length of the longest such line segment within all clusters.

Second, the maximum diameter resulting from each possible joining of two clusters is then calculated. The merging which leads to the smallest maximum diameter is chosen, and those two particular clusters become one.

The actual computer output of the cluster analysis describes the items included in each cluster at each stage of clustering. (One overall cluster, two clusters, three clusters, and so on up to n clusters are designated in succeeding stages.) Appendix M is a sample cluster analysis computer output.

The experimenter must decide how many clusters seem suitable to best describe the data groupings. This may be done a priori, but the decision is more often based on how well a given number of clusters describes the properties of the data. This rating may be determined by

$$\text{Strength Ratio} = N_{D\ \text{Within}}/N_{D\ \text{Total}}$$

where $N_{D\ \text{Within}}$ is the number of distances within clusters less than or equal to the maximum diameter, and

$N_{D\ \text{Total}}$ is the number of distances within and among clusters which is less than or equal to the maximum diameter.

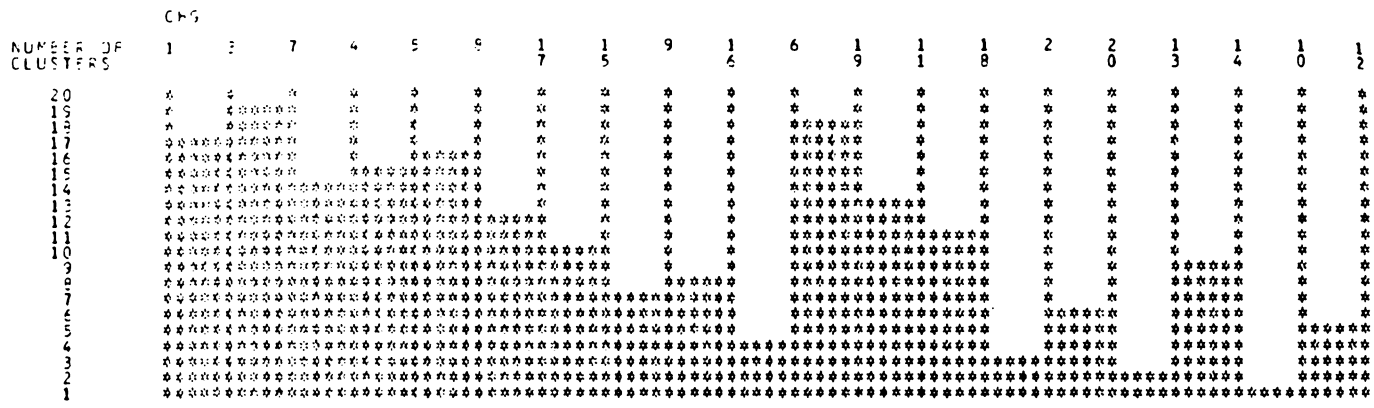
At the second stage of cluster analysis, the commands were clustered within each of these edit change clusters. Although all commands were clustered under each edit task group, within each group, some commands are not relevant in a given group. For example, the PASTE command cannot be used to insert or change characters or words. To limit the commands included in a given edit change cluster, two criteria were used. A cutoff was made at the point where the "strength" ratio decreased below .90. The second cutoff involved command relevance. Starting at the strongest cluster levels, a cutoff was drawn before the first irrelevant command, i.e., the first command which could not have been used to prepare for or actually make one of the edit changes in that edit change group. In each case, the cutoff which included fewer commands was taken.

Appendix M

SAMPLE CLUSTER ANALYSIS COMPUTER OUTPUT

CLUSTERING ACROSS ALL SUBJECTS ACROSS EDIT CHANGE

CLUSTER MAP



CLUSTERING ACROSS ALL SUBJECTS ACROSS EDIT CHANGE

CLUSTER ANALYSIS

NUMBER OF CLUSTERS	MAXIMUM DISTANCE WITHIN A CLUSTER	NUMBER OF DISTANCES WITHIN <= MAXIMUM	NUMBER OF DISTANCES IN ALL <= MAXIMUM	RATIO
20	0.00000000	0	0	0.00000
19	19.00000000	1	1	1.00000
18	31.00000000	2	3	0.66667
17	68.00000000	4	4	1.00000
16	99.00000000	5	5	1.00000
15	344.00000000	7	13	0.53846
14	390.00000000	16	16	1.00000
13	1506.00000000	18	26	0.69231
12	2125.00000000	24	48	0.50000
11	2230.00000000	27	54	0.50000
10	2540.00000000	34	71	0.47887
9	2740.00000000	35	82	0.42683
8	2978.00000000	36	87	0.41379
7	3086.00000000	52	91	0.57143
6	3346.00000000	53	95	0.55789
5	3525.00000000	54	103	0.52427
4	4322.00000000	94	141	0.66667
3	4827.00000000	122	151	0.80795
2	7088.00000000	154	177	0.87006
1	9981.00000000	190	190	1.00000

Appendix N

INSTRUCTIONS FOR FULL EDITOR

THE BASIC CONCEPTS OF INTERACTIVE TEXT EDITING

What you'll actually be doing for this experiment is fairly simple-- you'll be making editing corrections to a manuscript. This involves taking out parts of text, changing other parts, putting in new text, and so on. To do this, you'll use an interactive computer terminal with a keyboard and screen, using the text editor which is built into the system. You will communicate with the editing system through a terminal. A terminal looks like and sometimes behaves like a typewriter, except that it is connected to a computer. You tell the editing system, or editor, what you want it to do by typing commands at the terminal. The editor responds by performing your request, and both your typed input and the editor response are displayed on the terminal screen. Basically, the editor stores text as though it's on one long sheet of paper, but with only part of it displayed

at a time. In the editor you'll be working with, the "window", or the part of the text displayed at one time, is usually only a single line; the current line. When using most edit commands, knowing which line is the current line is very important.

When editing a manuscript, there are two modes. The first is called INPUT mode. When the editing system is in INPUT mode, anything that you type on the keyboard will appear on the screen exactly as you type it in, and it is stored on the electronic sheet of paper. This is much like the way that you add text to a blank piece of paper in a typewriter. The second mode is called EDIT mode. In this mode, what you type on the keyboard appears on the screen, but the editor interprets it as an editing command. When you are in EDIT mode, the editor will display a prompt (>) when it's ready for you to type in the next command. This prompt only appears in edit mode (not in input mode). You can switch back and forth between input and edit by entering a null carriage return. When you start editing, the editor will be in EDIT mode. Make sure you're in the right mode for what you want to do.

Before you start the actual editing, you'll become familar with some editing commands. These are the words or

phrases which you will type in at the terminal's keyboard, "commanding" the editor to do a specific operation. Instead of retyping or rewriting the manuscript by hand, you choose the commands you would type on the terminal keyboard to get the editor to perform the desired operation.

Following is a list first of the command names alone, then a description of all commands and their functions. After going through these descriptions, we'll discuss an example of the use of each command. But first, several things need to be noted:

1) The capitalized letters in each command name are an abbreviation for that command. For example, you may type in AR instead of the full AREA, or you could use DL for DELETE.

2) "n" refers to the number of lines over which the operation is to be performed. For example, to use the DELETE command to erase 3 lines, enter DELETE 3.

3) "string" represents a phrase or string of characters. For example, the CHANGE command could be used to change "oencil" to "pencil" for one line like this:

CHANGE 1 /oencil/pencil/

It is important to remember that the string you specify must be exactly like the string in the text. This includes using capital letters if needed.

4) You may enter commands in upper or lower case, the computer will understand both.

5) Make sure you put blank spaces in commands only where they are marked. For example, "CUTANDERASE" won't work if you type it as "CUT AND ERASE".

6) In commands where "n" is used, the default is 1. In other words, if you don't specify a number of lines ("n"), the editor will perform the operation on 1 line (the current line).

7) The line pointer is an (imaginary) plus sign which marks the current line. If the line pointer moves to a different line that line becomes the current line. The line pointer doesn't appear on the terminal screen, but is used in the command descriptions to show the location of the current line.

8) In both input and edit modes, you use the space bar to move forward, and the delete key to back up and erase previous characters within a line.

9) After typing in a command, you must enter a carriage return. Then the editor will "read" the command.

We will now go over the commands included in the editor.

Add

ARea

Bottom

CHange

ClearStore

Cut

CutandErase

Delete

DeLeteUp

DeLeteTO

Display

DisplayStore

Down

Find

FindUp

Insert

Join

Line

Mark

Paste

PasteandErase

REPeat

RETurn

SPlit

SWitch

Top

TRanspose

Up

Add string

Adds text to the end of an existing line (i.e. appends a string onto a line). Full line is displayed on the terminal screen. Total line must not exceed page width.

ARea

Displays a full screen of text around the current line, which remains at the same place. A plus sign in the left margin marks the current line.

Bottom

Moves the line pointer to the line below the last line of text, i.e. to the bottom of the manuscript. (Marked BOF for Bottom of File.)

CHange n,* /string1/string2/

Replaces unwanted text with wanted text by changing string1 to string2 for n lines. If * is used, instead of n, string1 is changed to string2 from the

current line to the bottom of the manuscript.

ClearStore n,* — Erases a specified number of lines which have been stored in the 'storage area' (starting with the first line stored). If * is used, all stored text is erased from the 'storage area'.

Cut n — Copies n lines, starting with the current line, into a storage area to be used later. The lines are not erased from the text manuscript itself.

CutandErase n,* — Places n lines, starting with the current line, into a storage area to be used later, and erases them from the text manuscript itself.

DeLete n,* — Erases n lines of text from the original manuscript. If a * is used, all text is deleted from the current line to the bottom of the manuscript.

DeLeteTO string — Erases the text from the beginning of the current line to the line containing the specified string.

DeLeteUP n, * — Deletes n lines, starting with the current line, up from the current line. If * is used, all text from the current line to the top of the file is deleted.

DISplay n,* — Displays a specified number of lines on the terminal screen, starting with the current line. The current line becomes the last line displayed. If * is used, all lines from the current line to the end of the manuscript are displayed. If no n or * are specified, the current line is displayed.

DisplayStore n,* — Displays n lines contained in the storage area (starting with the first line stored). If * is used, all lines stored are displayed.

Down n	Moves the line pointer down n lines from the present position, and displays the new current line.
Find string	The editor searches for a particular string of text from the current line to the bottom of the manuscript, moves the line pointer down to that line, and displays it on the screen. If the string is not found, the line pointer remains at the current line.
FindUp string	The editor searches for a particular string of text from the current line to the top of the manuscript, moves the line pointer up to that line, and displays it on the screen. If the string is not found, the line pointer remains at the current line.
Insert	Used for putting characters into the middle of an existing line. To use this command, enter 'insert', then a carriage return. Use the space bar to "write" the

line up to the place you will insert characters, then hit a carriage return. The editor will display the line up to that point. Type in the characters, then hit a carriage return.

Join — Combines the current line and the next line into one, which replaces the current line. The new line must not be wider than the page, and will be displayed on the screen.

Line — The editor responds to this by showing the line number of the current line.

Mark — Saves the location of the current line in memory so it can be returned to later after editing changes have been made elsewhere in the manuscript.

Paste n — Copies n lines from the storage area into the text after the current line. They are not erased from the storage area.

PasteandErase n,* Moves n lines from the storage area into the text after the current line and erases them from the storage area. Lines which have been in storage longest will be moved first.

REPeat Repeats the most recent successful command.

RETurn Returns to the last line specified by the Mark command.

SPlit string Breaks the current line into two lines, before the string. The entire second portion of the line need not be included in "string", just its beginning. Both lines will be displayed.

SWitch /word1/word2/

Exchanges the position of two words. The operations will be performed on the first found occurrence of each word1 and word2, starting with the current line.

Top — Moves the line pointer to the line above the first line of text, i.e. to the top of the manuscript. (Marked TOF for Top of File.)

TRanspose — Transposes the positions of two consecutive characters. To use this command, enter 'transpose' then a carriage return. Use the space bar to "write" out the line until you see the first of the two characters to be transposed and enter a carriage return. The new line will be displayed.

Up n — Moves the cursor upward the specified number of lines from the present position and displays the new current line on the screen.

EXAMPLES

So that you can see how each operation works, an example of each command follows. A sample file will be used to demonstrate the commands. Please go over the changes as I describe them. The text of the following examples shows you exactly what the terminal screen will display. The greater than sign (>) is the prompt which tells you the editor is ready for another command. The current line always appears just before the prompt.

At the top of each example is a copy of the full sample file as it appears before the command is entered. The current line is marked with a '+'. In most examples the relevant area of the sample file is also shown after the editing command has been issued (although you won't see this on the screen automatically.) If the text in the storage area is changed, the example shows the storage area contents before and after the change. Each example starts with the sample file as written below.

```
TOF:
This is
a test file
to be used
in examples.
It is to be
used for illustration
purposes.
BOF:
```

Add Command

```
  TOF
   This is
   a test file
 + to be used
   in examples.
   It is to be
   used for illustration
   purposes.
  BOF
```

```
Add string       to be used
                >add only

                 to be used only
                >
```

This is how the text will look after the command is issued:

```
   a test file
 + to be used only
   in examples.
```

Area Command

```
TOF
 This is
 a test file
 to be used
+ in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

ARea

```
 in examples
>area

TOF
 This is
 a test file
 to be used
+ in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

in examples.

>

Bottom Command

```
 TOF
  This is
+ a test file
  to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF
```

```
Bottom          a test file
               >bottom

                BOF
               >
```

This is how the text will look after the command is issued:

```
  It is to be
  used for illustration
  purposes.
+BOF
```

Change Command

```
 TOF
+ This is
  a test file
  to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF
```

```
CHange          This is
               >change * /is/was/

                This was
                It was to be
                2 lines changed

                This was
               >
```

This is how the text will look after the command is issued:

```
 TOF
+ This was
  a test file
  to be used
  in examples.
  It was to be
  used for illustration
  purposes.
 BOF
```

ClearStore Command

ClearStore

```
It is to be
>clearstore *

All stored lines deleted.

It is to be
>
```

Storage Area Before:

```
a test file
to be used
```

Storage Area After:

Cut Command

```
TOF
 This is
 a test file
+ to be used
 in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

Cut

```
 to be used
>cut 2

 to be used
>
```

This is how the text will look after the command is issued:

```
 a test file
+ to be used
 in examples.
```

Storage Area Before:

Storage Area After:

to be used

in examples.

CutandErase Command

```
 TOF
+ This is
  a test file
  to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF
```

CutandErase

```
 This is
>cutanderase 3

 in examples
>
```

This is how the text will look after the command is issued:

```
 TOF
+ in examples
  It is to be
```

Storage Area Before:

Storage Area After:

```
This is
a test file
to be used
```

Delete Command

```
TOF
 This is
 a test file
 to be used
 in examples.
+It is to be
 used for illustration
 purposes.
BOF
```

```
DeLete          It is to be
               >delete 2

                purposes.
               >
```

This is how the text will look after the command is issued:

```
 to be used
 in examples
+purposes.
BOF
```

Deleteto Command

```
 TOF
+ This is
  a test file
  to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF
```

```
DeLeteTO          This is
                 >deleteto used

                  to be used
                 >
```

This is how the text will look after the command is issued:

```
 TOF
+ to be used
  in examples.
```

Deleteup Command

```
 TOF
  This is
  _ test file
  to be used
  in examples.
  It is to be
 + used for illustration
  purposes.
 BOF
```

```
DeLeteUP          used for illustration
                 >deleteup 3

                  to be used
                 >
```

This is how the text will look after the command is issued:

```
                + to be used
                  purposes.
                 BOF
```

Display Command

```
TOF
 This is
+ a test file
 to be used
 in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

Display

```
 a test file
>display 2

 a test file
 to be used
>
```

This is how the text will look after the command is issued:

```
 a test file
+ to be used
 in examples.
```

DisplayStore Command

```
DisplayStore      a test file
                 >displaystore 2

                  first line
                  second line

                  a test file
                 >
```

Storage Area Before:

```
                    first line
                    second line
```

Storage Area After:

```
                    first line
                    second line
```

Down Command

```
TOF
 This is
 a test file
 to be used
 in examples.
 It is to be
+used for illustration
 purposes.
 BOF

Down          used for illustration
             >down 2

             BOF
             >
```

This is how the text will look after the command is issued:

```
 used for illustration
 purposes.
+BOF
```

Find Command

```
TOF
 This is
 a test file
 to be used
 in examples.
+It is to be
 used for illustration
 purposes.
BOF
```

```
Find          to be used
             >find It

              It is to be
             >
```

This is how the text will look after the command is issued:

```
              to be used
              in examples.
             +It is to be
```

Findup Command

```
TOF
 This is
 a test file
 to be used
 in examples.
 It is to be
 used for illustration
+ purposes.
 BOF

FindUp          purposes
               >findup in

                in examples.
               >
```

This is how the text will look after the command is issued:

```
               + in examples.
                 It is to be
                 used for illustration
                 purposes.
```

Insert Command

```
TOF
 This is
 a test file
 to be used
+ in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

Insert in examples.

```
>insert
(press space bar 2 times to produce:)
 >in
```

Type in 'several'

```
>in several

 in several
>
```

This is how the text will look after the command is issued:

to be used

+ in several examples.

It is to be

Join Command

```
TOF
 This is
 a test file
+ to be used
 in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

```
Join            to be used
               >join

                to be used in examples.
               >
```

This is how the text will look after the command is issued:

```
 a test file
+ to be used in examples
 It is to be
```

Line Command

Line to be used

```
>line

 The current line is 3

 to be used
>
```

Mark Command and Return Command

Mark and Return This is

>mark

This is

>down 3

in examples.

>return

This is

>

Paste Command

```
TOF
 This is
 a test file
 to be used
+ in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

Paste

```
 in examples.
>paste 2

 second line
>
```

This is how the text will look after the command is issued:

```
 in examples.
 first line
+ second line
 It is to be
```

Storage Area Before:

first line

second line

Storage Area After:

first line

second line

PasteandErase Command

```
TOF
  This is
  a test file
  to be used
+ in examples.
  It is to be
  used for illustration
  purposes.
BOF

PasteandErase      in examples.
                  >pasteanderase 2

                    second line
                  >
```

This is how the text will look after the command is issued:

```
                    in examples.
                    first line
                  + second line
                    It is to be
```

Storage Area Before:

```
first line
second line
```

Storage Area After:

Repeat Command

```
 TOF
+ This is
  a test file
  to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF

REPeat          This is
               >down 1

                a test file
               >repeat

                to be used
               >
```

This is how the text will look after the command is issued:

```
                This is
```

```
  a test file
+ to be used
  in examples.
```

RETurn	See Mark

Split Command

```
                TOF
                 This is
                 a test file
                + to be used
                 in examples.
                 It is to be
                 used for illustration
                 purposes.
                BOF

SPlit            to be used
                >split be

                 be used
                >
```

This is how the text will look after the command is issued:

```
                 a test file
                 to
                + be used
                 in examples.
```

Switch Command

```
TOF
 This is
+ a test file
 to be used
 in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

SWitch

```
 a test file
>switch /test/file/

 a file test
>
```

This is how the text will look after the command is issued:

```
 This is
+ a file test
 to be used
```

```
                TOF
                 This is
                 a test file
               + to be used
                 in examples.
                 It is to be
                 used for illustration
                 purposes.
                BOF

Top              to be used
                >top

                TOF
                >
```

This is how the text will look after the command is issued:

```
               + TOF
                 This is
                 a test file
```

Transpose Command

```
 TOF
  This is
  a test file
+ to be used
  in examples.
  It is to be
  used for illustration
  purposes.
 BOF
```

```
TRanspose        to be used
                 > transpose
                 > (press space bar 7 times to get:)
                 >to be u

                  to be sued
                 >
```

This is how the text will look after the command is issued:

```
  a test file
+ to be sued
```

in examples.

Up Command

```
TOF
 This is
 a test file
 to be used
+ in examples.
 It is to be
 used for illustration
 purposes.
BOF
```

```
Up              in examples
               >up 2

                a test file
               >
```

This is how the text will look after the command is issued:

```
 This is
+ a test file
 to be used
 in examples.
```

This completes the descriptions and examples of the editing commands you will be using.

Appendix O

COMPLETE SUMMARY TABLE--INITIAL AND SUPPLEMENTAL ANALYSES

PRIMARY ANOVA

Dependent Variable: TC Frequency count of commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	34603.81	8.73	0.0058
EDITOR (E)	3	3261.22	0.27	0.8434
U*E	3	2697.47	0.23	0.8770
Subj/UE	32	126799.60		
Within-Subject				
PRACTICE (P)	1	1171.81	0.90	0.3506
U*P	1	529.26	0.41	0.5289
E*P	3	1235.87	0.32	0.8141
U*E*P	3	3064.62	0.78	0.5126
P*Subj/UE	32	41788.20		
TEXT (T)	1	232.81	0.23	0.6374
U*T	1	1182.66	1.15	0.2916
E*T	3	8340.37	2.70	0.0619
U*E*T	3	1306.52	0.42	0.7374
T*Subj/UE	32	32906.40		
T*P	1	14118.81	5.89	0.0210
U*T*P	1	6187.65	2.58	0.1179
E*T*P	3	3399.37	0.47	0.7034
U*E*T*P	3	3828.52	0.53	0.6633
T*P*Subj/UE	32	76697.40		

Dependent Variable: TE Frequency count of errors

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	10.50625000	0.41	0.5250
EDITOR (E)	3	48.31875000	0.63	0.5990
U*E	3	34.11875000	0.45	0.7210
Subj/UE	32	813.90000000		
Within-Subject				
PRACTICE (P)	1	15.00625000	2.17	0.1503
U*P	1	15.00625000	2.17	0.1503
E*P	3	31.01875000	1.50	0.2342
U*E*P	3	31.11875000	1.50	0.2330
P*Subj/UE	32	221.10000000		
TEXT (T)	1	16.25625000	2.31	0.1383
U*T	1	0.50625000	0.07	0.7902
E*T	3	21.76875000	1.03	0.6828
U*E*T	3	10.61875000	0.50	0.6828
T*Subj/UE	32	225.10000000		
T*P	1	91.50625000	3.85	0.0585
U*T*P	1	4.55625000	0.19	0.6645
E*T*P	3	87.61875000	1.23	0.3154
U*E*T*P	3	49.86875000	0.70	0.5594
T*P*Subj/UE	32	760.70000000		

Dependent Variable: TT Time (including 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	4450897381.2249984	77.16	0.0001
EDITOR (E)	3	583400.6499996	0.00	0.9997
U*E	3	416179301.0249987	2.40	0.0856
Subj/UE	32	1845927377.7000045		
Within-Subject				
PRACTICE (P)	1	971249670.4000005	28.21	0.0001
U*P	1	1822009.2250004	0.05	0.8195
E*P	3	59468587.8499966	0.58	0.6352
U*E*P	3	73146897.8250008	0.71	0.5543
P*Subj/UE	32	1101901752.7000017		
TEXT (T)	1	6287697.0250006	0.32	0.5774
U*T	1	18114468.1000004	0.91	0.3465
E*T	3	136508912.3249969	2.29	0.0967
U*E*T	3	99821264.6500006	1.68	0.1915
T*Subj/UE	32	634852955.9000015		
T*P	1	156899171.0249977	2.01	0.1660
U*T*P	1	44846532.8999987	0.57	0.4541
E*T*P	3	20394190.3250027	0.09	0.9666
U*E*T*P	3	297154366.6499958	1.27	0.3018
T*P*Subj/UE	32	2498980876.1000051		

Dependent Variable: T Time (not incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	3318663758.3999967	58.99	0.0001
EDITOR (E)	3	16386048.6499977	0.10	0.9611
U*E	3	379291272.7500019	2.25	0.1018
Subj/UE	32	1800196361.7000036		
Within-Subject				
PRACTICE (P)	1	917974772.0999984	39.85	0.0001
U*P	1	758451.6000023	0.03	0.8572
E*P	3	28281247.5499992	0.41	0.7474
U*E*P	3	86704153.8499975	1.25	0.3064
P*Subj/UE	32	737091794.9000024		
TEXT (T)	1	14401200.0249977	0.54	0.4678
U*T	1	15713876.0250015	0.59	0.4484
E*T	3	116649114.7249985	1.46	0.2445
U*E*T	3	74771799.7249994	0.93	0.4354
T*Subj/UE	32	853532757.5000028		
T*P	1	40684907.0250025	0.56	0.4605
U*T*P	1	6416811.0249968	0.09	0.7686
E*T*P	3	41702575.0250006	0.19	0.9020
U*E*T*P	3	197353445.8249960	0.90	0.4508
T*P*Subj/UE	32	2332846855.1000041		

ANOVA OF DATA SET WITHOUT 'CHECKING'

Dependent Variable: TC Frequency count of commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	39344.25625	9.75	0.0038
EDITOR (E)	3	2590.86875	0.21	0.8860
U*E	3	1772.21875	0.15	0.9313
Subj/UE	32	129189.10000		
Within-Subject				
PRACTICE (P)	1	743.90625	0.54	0.4687
U*P	1	636.00625	0.46	0.5026
E*P	3	941.86875	0.23	0.8770
U*E*P	3	3845.16875	0.93	0.4392
P*Subj/UE	32	44274.30000		
TEXT (T)	1	142.50625	0.13	0.7223
U*T	1	1086.80625	0.98	0.3295
E*T	3	7079.46875	2.13	0.1160
U*E*T	3	1293.56875	0.39	0.7617
T*Subj/UE	32	35475.90000		
T*P	1	12906.05625	5.69	0.0232
U*T*P	1	6464.30625	2.85	0.1012
E*T*P	3	4930.81875	0.72	0.5449
U*E*T*P	3	5719.76875	0.84	0.4819
T*P*Subj/UE	32	72606.30000		

Dependent Variable: TE Frequency count of errors

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	11.025	0.43	0.5160
EDITOR (E)	3	47.925	0.63	0.6040
U*E	3	33.125	0.43	0.7324
Subj/UE	32	817.800		
Within-Subject				
PRACTICE (P)	1	14.400	2.09	0.1577
U*P	1	14.400	2.09	0.1577
E*P	3	29.850	1.45	0.2478
U*E*P	3	30.150	1.46	0.2438
P*Subj/UE	32	220.200		
TEXT (T)	1	16.900	2.39	0.1319
U*T	1	0.400	0.06	0.8135
E*T	3	22.850	1.08	0.3725
U*E*T	3	10.650	0.50	0.6835
T*Subj/UE	32	226.200		
T*P	1	93.025	3.92	0.0564
U*T*P	1	4.225	0.18	0.6759
E*T*P	3	89.225	1.23	0.3069
U*E*T*P	3	51.125	0.72	0.5484
T*P*Subj/UE	32	759.400		

Dependent Variable: TT Time (including 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	4704192277.2249980	78.95	0.0001
EDITOR (E)	3	459219.1499996	0.00	0.9998
U*E	3	385900305.0249996	2.16	0.1122
Subj/UE	32	1906741937.7000036		
Within-Subject				
PRACTICE (P)	1	877257024.4000005	24.65	0.0001
U*P	1	3774259.2250004	0.11	0.7468
E*P	3	68796409.8499966	0.64	0.5922
U*E*P	3	102050068.8249998	0.96	0.4254
P*Subj/UE	32	1138775030.7000026		
TEXT (T)	1	5639259.0250006	0.27	0.6056
U*T	1	20920729.6000004	1.01	0.3227
E*T	3	106661500.3249969	1.71	0.2836
U*E*T	3	93480782.1499996	1.50	0.2325
T*Subj/UE	32	663462071.9000024		
T*P	1	152298965.0249977	2.17	0.1500
U*T*P	1	39231724.8999987	0.56	0.4596
E*T*P	3	20657903.3250027	0.10	0.9604
U*E*T*P	3	217602850.6499968	1.04	0.3899
T*P*Subj/UE	32	2240736978.1000041		

Dependent Variable: T Time (not incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	3200986156.8999986	39.67	0.0001
EDITOR (E)	3	49510056.1250000	0.20	0.8925
U*E	3	272715077.3499966	2.25	0.1018
Subj/UE	32	2582141037.4000062		
Within-Subject				
PRACTICE (P)	1	410970744.8999996	10.43	0.0029
U*P	1	29242710.0250015	0.74	0.3955
E*P	3	82490260.0499983	0.70	0.5604
U*E*P	3	57362271.8250008	0.49	0.6950
P*Subj/UE	32	737091794.9000024		
TEXT (T)	1	12025315.6000004	0.29	0.5961
U*T	1	16740478.2249985	0.40	0.5321
E*T	3	303068950.8499975	2.41	0.0854
U*E*T	3	181500987.7250032	1.44	0.2489
T*Subj/UE	32	1342615807.6000003		
T*P	1	28586355.6000004	0.35	0.5569
U*T*P	1	52684020.8999996	0.65	0.4263
E*T*P	3	124281638.0250006	0.51	0.6778
U*E*T*P	3	229619284.0499945	0.94	0.4312
T*P*Subj/UE	32	2595859252.4000053		

ANOVA LIMITED TO NOVICE AND EXPERT MODEL EDITOR CONDITIONS

Dependent Variable: TC Frequency count of commands

SOURCE	df	SS	F	p
Between-Subject				
USER (U)	1	11544.0125	5.69	0.0298
EDITOR (E)	1	2542.5125	1.25	0.2797
U*E	1	208.0125	0.10	0.7531
Subj/UE	16	32488.1000		
Within-Subject				
PRACTICE (P)	1	891.1125	1.47	0.2429
U*P	1	63.0125	0.10	0.7513
E*P	1	1015.3125	1.68	0.2319
U*E*P	1	49.6125	0.08	0.7785
P*Subj/UE	16	9697.7000		
TEXT (T)	1	1419.6125	2.63	0.1241
U*T	1	1419.6125	2.63	0.1241
E*T	1	1178.1125	2.19	0.1586
U*E*T	1	690.3125	1.28	0.2744
T*Subj/UE	16	8621.1000		
T*P	1	14124.6125	10.69	0.0048
U*T*P	1	3795.0125	2.87	0.1095
E*T*P	1	959.1125	0.73	0.4068
U*E*T*P	1	556.5125	0.42	0.5256
T*P*Subj/UE	16	21145.5000		

Dependent Variable: TE		Frequency count of errors		
SOURCE	df	SS	F	p
Between-Subject				
USER (U)	1	3.20	0.21	0.6517
EDITOR (E)	1	0.80	0.05	0.8210
U*E	1	0.80	0.05	0.8210
Subj/UE	16	241.90		
Within-Subject				
PRACTICE (P)	1	18.05	2.06	0.1709
U*P	1	4.05	0.46	0.5068
E*P	1	14.45	1.65	0.2178
U*E*P	1	26.45	3.01	0.1019
P*Subj/UE	16	140.50		
TEXT (T)	1	14.45	1.56	0.2295
U*T	1	0.05	0.01	0.9423
E*T	1	14.45	1.56	0.2295
U*E*T	1	0.45	0.05	0.8283
T*Subj/UE	16	148.10		
T*P	1	33.80	3.06	0.0994
U*T*P	1	7.20	0.65	0.4313
E*T*P	1	64.80	5.87	0.0277
U*E*T*P	1	45.00	4.07	0.0606
T*P*Subj/UE	16	176.70		

Dependent Variable: TT Time (incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subject				
USER (U)	1	2709407893.61249920	39.99	0.0001
EDITOR (E)	1	277.51249886	0.00	0.9984
U*E	1	87401714.51249980	1.29	0.2728
Subj/UE	16	1084038031.80000200		
Within-Subject				
PRACTICE (P)	1	671820157.01249880	44.78	0.0001
U*P	1	3055665.31249905	0.20	0.6578
E*P	1	28815602.11250019	1.92	0.1848
U*E*P	1	1236785.11250114	0.08	0.7777
P*Subj/UE	16	240027679.20000076		
TEXT (T)	1	28071466.51249790	1.43	0.2497
U*T	1	44439257.81250000	2.26	0.1523
E*T	1	7082285.11250210	0.36	0.5569
U*E*T	1	73044153.11249923	3.71	0.0719
T*Subj/UE	16	314744259.20000076		
T*P	1	37929465.31250286	0.36	0.5594
U*T*P	1	42202387.81249905	0.40	0.5383
E*T*P	1	4852602.61249542	0.05	0.8338
U*E*T*P	1	18515614.61250114	0.17	0.6826
T*P*Subj/UE	16	1707539381.40000150		

Dependent Variable: T Time (not incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subject				
USER (U)	1	2032470734.44999790	30.14	0.0001
EDITOR (E)	1	4983014.44999886	0.07	0.7892
U*E	1	135257206.05000209	2.01	0.1759
Subj/UE	16	1078945636.30000110		
Within-Subject				
PRACTICE (P)	1	479964828.79999920	28.86	0.0001
U*P	1	622339.20000076	0.04	0.8491
E*P	1	19621804.99999905	1.18	0.2935
U*E*P	1	5637096.19999886	0.34	0.5686
P*Subj/UE	16	266124420.30000209		
TEXT (T)	1	1625070.04999828	0.05	0.8261
U*T	1	6401461.25000191	0.20	0.6636
E*T	1	2046080.44999981	0.06	0.8054
U*E*T	1	56525306.44999790	1.73	0.2064
T*Subj/UE	16	521538036.30000200		
T*P	1	2040.20000076	0.00	0.9967
U*T*P	1	4433052.79999924	0.04	0.8642
E*T*P	1	3075.20000076	0.00	0.9959
U*E*T*P	1	7021125.00000000	0.06	0.8072
T*P*Subj/UE	16	1825185266.29999920		

EXPERT SUBJECTS' DATA ONLY

Dependent Variable: TC Frequency count of commands

SOURCE	df	SS	F	p
Between-Subjects				
EDITOR (E)	3	2768.95	1.93	0.7370
Subj/E	16	34666.10		
Within-Subject				
PRACTICE (P)	1	1638.05	1.93	0.1840
E*P	3	2638.95	1.04	0.4036
P*Subj/E	16	13596.50		
TEXT (T)	1	1232.45	1.83	0.1954
E*T	3	3425.75	1.69	0.2088
T*Subj/E	16	10800.30		
T*P	1	806.45	0.40	0.5360
E*T*P	3	4142.95	0.69	0.5741
T*P*Subj/E	16	32249.10		

Dependent Variable: TE		Frequency count of errors		
SOURCE	df	SS	F	p
Between-Subjects				
EDITOR (E)	3	62.1375	1.31	0.3049
Subj/E	16	252.5000		
Within-Subject				
PRACTICE (P)	1	30.0125	6.97	0.0178
E*P	3	41.8375	3.24	0.0500
P*Subj/E	16	68.9000		
TEXT (T)	1	5.5125	0.94	0.3479
E*T	3	7.9375	0.45	0.7215
T*Subj/E	16	94.3000		
T*P	1	27.6125	1.34	0.2639
E*T*P	3	131.6375	2.13	0.1364
T*P*Subj/E	16	32249.10		

Dependent Variable: TT Time (incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
EDITOR (E)	3	215879876.43749904	1.19	0.3466
Subj/E	16	971332890.70000170		
Within-Subject				
PRACTICE (P)	1	528602761.01249980	31.90	0.0001
E*P	3	45918636.33749961	0.92	0.4519
P*Subj/E	16	265159793.90000057		
TEXT (T)	1	22873396.61250019	1.53	0.2343
E*T	3	42712636.53749943	0.95	0.4396
T*Subj/E	16	239609933.10000038		
T*P	1	16989696.11249828	1.20	0.2889
E*T*P	3	142477112.43749809	3.36	0.0450
T*P*Subj/E	16	225954856.70000362		

Dependent Variable: T Time (not incl. 'fixing')

SOURCE	df	SS	F	p
Between-Subjects				
EDITOR (E)	3	220476480.84999942	1.23	0.3320
Subj/E	16	957632445.70000170		
Within-Subject				
PRACTICE (P)	1	485752961.25000000	26.48	0.0001
E*P	3	42418773.24999905	0.77	0.5271
P*Subj/E	16	293561173.50000095		
TEXT (T)	1	30100764.80000019	1.96	0.1811
E*T	3	48297047.49999904	1.05	0.3992
T*Subj/E	16	246263937.70000076		
T*P	1	7393279.99999905	0.54	0.4739
E*T*P	3	128343192.69999790	3.11	0.0557
T*P*Subj/E	16	219879776.30000305		

RATIO OF BASIC TO TOTAL COMMANDS ANOVA

Dependent Variable: B Ratio of basic to total commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	0.21161629	18.57	0.0001
EDITOR (E)	3	0.38319869	11.21	0.0001
U*E	3	0.09826027	2.87	0.0515
Subj/UE	32	0.36470091		
Within-Subject				
PRACTICE (P)	1	0.00026414	0.30	0.5876
U*P	1	0.00108366	1.23	0.2754
E*P	3	0.00335430	1.27	0.3010
U*E*P	3	0.00029290	0.11	0.9531
P*Subj/UE	32	0.02815855		
TEXT (T)	1	0.00001107	0.01	0.9258
U*T	1	0.00008794	0.07	0.7930
E*T	3	0.00186611	0.50	0.6882
U*E*T	3	0.00164025	0.44	0.7293
T*Subj/UE	32	0.04020152		
T*P	1	0.00079472	0.06	0.8055
U*T*P	1	0.00153701	0.12	0.7321
E*T*P	3	0.04539306	1.17	0.3350
U*E*T*P	3	0.01942600	0.50	0.6833
T*P*Subj/UE	32	0.41243198		

NOVICE COMMAND SET--ACROSS MODEL EDITORS ONLY

Dependent Variable: NC Frequency count of command subset

SOURCE	df	SS	F	p
Between-Subject				
USER (U)	1	12751.25	5.97	0.0265
EDITOR (E)	1	5152.05	0.09	0.1399
U*E	1	460.80	0.22	0.6485
Subj/UE	16	34159.60		
Within-Subject				
PRACTICE (P)	1	1022.45	1.73	0.2071
U*P	1	51.20	0.09	0.7724
E*P	1	897.80	1.52	0.2357
U*E*P	1	26.45	0.04	0.8352
P*Subj/UE	16	9461.60		
TEXT (T)	1	1428.05	2.60	0.1261
U*T	1	1411.20	2.57	0.1282
E*T	1	1248.20	2.28	0.1509
U*E*T	1	708.05	1.29	0.2726
T*Subj/UE	16	8774.00		
T*P	1	13005.00	9.69	0.0067
U*T*P	1	3406.05	2.54	0.1307
E*T*P	1	1170.45	0.87	0.3643
U*E*T*P	1	405.00	0.30	0.5904
T*P*Subj/UE	16	21474.00		

ANOVA OF ALTERING AND MOVING COMMANDS AND THEIR RATIO

Dependent Variable: A Frequency count of altering commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	543.90625	3.24	0.0813
EDITOR (E)	3	485.31875	0.96	0.4217
U*E	3	123.66875	0.25	0.8638
Subj/UE	32	5370.70000		
Within-Subject				
PRACTICE (P)	1	1.05625	0.02	0.8791
U*P	1	51.75625	1.15	0.2913
E*P	3	39.71875	0.29	0.8291
U*E*P	3	352.51875	2.61	0.0682
P*Subj/UE	32	1438.70000		
TEXT (T)	1	327.75625	8.58	0.0062
U*T	1	43.05625	1.13	0.2963
E*T	3	527.31875	4.60	0.0087
U*E*T	3	253.51875	2.21	0.1057
T*Subj/UE	32	1222.10000		
T*P	1	860.25625	5.89	0.0211
U*T*P	1	91.50625	0.63	0.2248
E*T*P	3	672.11875	1.53	0.2248
U*E*T*P	3	406.56875	0.93	0.4387
T*P*Subj/UE	32	4675.30000		

Dependent Variable: M Frequency count of moving commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	9105.30625	5.39	0.0268
EDITOR (E)	3	1818.16875	0.36	0.7831
U*E	3	1207.16875	0.24	0.8690
Subj/UE	32	54050.30000		
Within-Subject				
PRACTICE (P)	1	985.05625	1.44	0.2389
U*P	1	465.80625	0.68	0.4154
E*P	3	914.71875	0.45	0.7220
U*E*P	3	2408.16875	1.17	0.3352
P*Subj/UE	32	21888.50000		
TEXT (T)	1	1683.50625	2.73	0.1082
U*T	1	432.30625	0.70	0.4085
E*T	3	5662.36875	3.06	0.0421
U*E*T	3	635.36875	0.34	0.7937
T*Subj/UE	32	19721.70000		
T*P	1	8079.80625	5.73	0.0227
U*T*P	1	4526.25625	3.21	0.0827
E*T*P	3	2755.76875	0.65	0.5881
U*E*T*P	3	781.51875	0.18	0.9060
T*P*Subj/UE	32	45143.90000		

Dependent Variable: A/M Ratio of alter to move commands

SOURCE	df	SS	F	p
Between-Subjects				
USER (U)	1	0.11821207	0.44	0.5128
EDITOR (E)	3	0.35475545	0.44	0.7272
U*E	3	0.29137835	0.36	0.7823
Subj/UE	32	8.63454945		
Within-Subject				
PRACTICE (P)	1	0.13780960	1.24	0.2733
U*P	1	0.01511170	0.14	0.7145
E*P	3	0.83609656	2.51	0.0761
U*E*P	3	0.21755961	0.65	0.5864
P*Subj/UE	32	3.54893292		
TEXT (T)	1	0.35524860	5.83	0.0216
U*T	1	0.00370662	0.06	0.8067
E*T	3	0.86956087	4.76	0.0074
U*E*T	3	0.18667548	1.02	0.3959
T*Subj/UE	32	1.94859165		
T*P	1	0.48272533	2.36	0.1347
U*T*P	1	0.38741116	1.89	0.1787
E*T*P	3	0.93935047	1.53	0.2262
U*E*T*P	3	0.13435304	0.22	0.8828
T*P*Subj/UE	32	1.94859165		

REFERENCES

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

Card, S. K., Moran, T. P., and Newell, A. The keystroke level model for user performance time with interactive systems. Xerox Report SSL-79-1, March, 1979.

Cohill, A. M. Unpublished Master's Thesis, A study of information presentation in software HELP systems. Virginia Polytechnic Institute and State University, December, 1981.

Dumais, S. T. and Landauer, T. K. Psychological investigations of natural command & query terminology. Bell Laboratories, Murray Hill, NJ, 1981.

Ehrich, R. W. SAM. Developed in the Computer Science Department of Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 24, 1980.

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, 10 (6), Nov. 1978, 639-650.

Embley, D. W. and Nagy, G. Behavioral aspects of text editors. Computing Surveys, 13 (1), March 1981, 33-70.

Embley, D. W. and Nagy, G. Methods for the study of computer editors. In M. J. Coombs and J.L. Alty (Eds.) Computer skills and the user interface. New York: Academic Press, 1981.

Hammer, J. M. The human as a constrained optimal editor. PhD Thesis, University of Illinois, 1981.

Jackson, M.D. and Tschirgi, J. E. The nature of user generated commands for interacting with a computer. Proceedings of the International Conference on Cybernetics and Society, 1981, 29-32.

Johnson, S. C. Hierarchical clustering schemes, Psychometrika, XXXII, 1967, 241-254.

Ledgard, H. and Whiteside, J. A. The natural language of interactive systems. Communications of the ACM, 1980, 23, (10), 556-563.

Mayer, R. E. and Bayman, P. Psychology of calculator languages: A framework for describing differences in users' knowledge. Communications of the ACM. August 1981, 24, (8), 511-520.

Miller, L.A. Natural language programming: Styles, strategies and contrasts. IBM Systems Journal, 1981, 20, (2), 184-215.

Roberts, T. L. Evaluation of computer text editors. Xerox Report SSL-79-9, November, 1979.

Robertson, C. K., McCracken, D.L. and Newell, A. Experimental evaluation of the ZOG frame editor. Proceedings of the Conference of Canadian Man-Computer Communications Society, Waterloo, Ontario, Canada, June 10-12, 1981.

Scapin, D. L. Computer commands in restricted natural language: Some aspects of memory and experience. Human Factors, 1981, 23 (3), 365-375.

Thomas, J. C. and Carroll, J. M. Human factors in communication. IBM Systems Journal, 1981, 20 (2), 237-263.

The two page vita has been removed from the scanned document. Page 1 of 2

The two page vita has been removed from the scanned document. Page 2 of 2

THE DEVELOPMENT AND VALIDATION OF A USER'S MODEL FOR INTERACTIVE TEXT EDITING

by

Lisa Joanne Folley

(ABSTRACT)

Interactive computer editors are an integral part of today's growing computer systems. In spite of this, their design is often determined by tradition or designer intuition, rather than empirical evidence. These studies explore a method of deriving naive and expert user models by use of a clustering algorithm. The models, which consisted of a core set of commands, were developed by applying a heirarchical cluster analysis to the paper and pencil responses of naive and expert subjects. It was expected that in an actual interactive editing environment, performance would be better when the operator used the appropriate model editor.

A validation study tested the models in an actual interactive editing environment. Analysis of variance results showed that novice and expert subjects did not have fewer errors, use fewer commands, or take less time to edit when using an editor based on the appropriate user model. In general, user (novice and expert), practice (first or second editing session) and some higher order interactions were the

only significant effects. It was hypothesized that the high heterogeneity of variance might be obscuring some editor differences. A cluster analysis performed on the validation study data showed low agreement between subjects and with the original models. Differences in command frequency counts between the paper-pencil and interactive editing environments suggest that deeper aspects of interactive editing must be included in the user models. Not only command set, but mode change and current line location must be considered. It is suggested that interactive editors will more closely conform to user models as they more closely simulate the paper-pencil editing environment.