Behavioral Aspects of Text Editors

DAVID W. EMBLEY AND GEORGE NAGY

University of Nebraska, Lincoln, Nebraska 68588

Theoretical models and experimental results relevant to the study of behavioral issues in the use of text editors—including both those intended primarily for computer program development and those intended for manuscript preparation—are examined. Models can predict editing task time in terms of elementary activities, in an error-free environment, to an accuracy comparable to the variability between subjects. In a realistic setting, however, unpredictable user activities account for between 25 and 50 percent of the task time, an amount that is comparable to individual variations due to errors. Variations in computer response time appear to affect users more than mere delay does. Command options improve expert performance but degrade the performance of beginners. The surface syntax of an editor can have considerable impact on ease of use. Ergonomic aspects of keyboard and display terminal design and use are well understood, with little hope for significant improvement, but there is no experimental evidence to support guidelines for display format design. Among analog pointing devices the mouse appears to have a small edge over the light pen, joystick, and track ball; human pointing performance using these devices approaches known psychophysical limits. Optimum ambient conditions, including temperature, noise, work-station layout, illumination, and work-rest cycles derived for professional key entry operators and for other interactive tasks, are probably also valid for editing. Gaps in the application of cognitive psychology and human engineering to text editors in the literature are indicated, and promising research areas are delineated.

Keywords and Phrases: text editors, program editors, manuscript editors, controlled experiments, behavioral science, cognitive psychology, human factors, keyboards, data entry, behavioral models, interactive input

CR Categories: 1.3, 4.4, 4.6

INTRODUCTION

The objective of this paper is to give an account of the methods in use for studying the behavioral aspects of text editors and to present the results obtained. In the introduction we briefly describe the various functions of computer editors, discuss diverse means of studying them, and provide pointers to applicable areas of psychology. It is assumed that the reader is familiar with the basic vocabulary of computer science, has had sufficient exposure to various text and program editors to have built up firm opinions regarding them, and is innocent of any formal training in psychology.

For background reading in the more comprehensive area of interactive systems, we recommend the Infotech report on "Man/Computer Communications" (with a bibliography of 225 titles) [INFO79], Martin's popular *Design of Man/Computer Dialogues* [MART73], and the *International Journal of Man-Machine Studies.*

CONTENTS

INTRODUCTION
Interactive Text Editors
Editor Design and Evaluation
Applicable Areas of Psychology
1. PERFORMANCE TIME CONSIDERATIONS
1.1 Simple Models for Predicting Task Time
1.2 More Comprehensive Models
1.3 Observations of Overall Task Time
1.4 The Effects of Computer Response Time
2. EDITOR DESIGN CONSIDERATIONS
2.1 Popular Wisdom
2.2 Observation
2.3 Syntactic Analysis
2.4 Controlled Experiments
3. INPUT AND OUTPUT DEVICES AND TECHNIQUES
3.1 Key Entry
3.2 Display Devices and Screen-Oriented Operations
4. CONCLUSIONS
4.1 Outstanding Design Issues
4.2 Summary of Results
ACKNOWLEDGMENTS
REFERENCES

Interactive Text Editors

Interactive text editors allow the manipulation of a set of files stored on the host machine by means of a terminal device such as a teletypewriter or a display-keyboard combination. The files may contain natural language text, computer programs, or alphanumeric data. Because editors frequently constitute the primary means of interaction of a person with the computer, they tend to subsume all kinds of secondary functions as well.

In a narrow sense one may consider the editing process as a transformation from an existing string of symbols known as the source file (which, in the case of initial text entry, may be null) to a new string of symbols known as the target file [OREN74, HECK78, ANAN80]. This definition must be stretched to accommodate hierarchical editors, which also contain information about the semantic structure of the files; formatting editors, which change margins, justify text, center headers, and provide other typesetting commands; and language-dependent editors, which accommodate the syntactic rules of a programming language (or, eventually, even of natural language!). An even broader interpretation is necessary to include file organization, message systems, and access to utility programs. Editors often also provide status information, such as the number of current users or the time of the day, and access to the output of compilers, interpreters, and other programs submitted for execution.

A thorough survey of interactive editors appears in VAND71, with more recent contributions referenced in RIDD76 and REIM78. Widely known examples of general-purpose interactive editors are QED, CMS, TECO, Wylbur, WIDJET, and UNIX [DEUT67, IBM76, TECO69, STAN75, IBM78, KERN79]. Unlike general-purpose editors, language-dependent editors are restricted to program modification in some specified programming language. Therefore, they can incorporate syntax validation, diagnostic messages, and error-correcting functions normally reserved for compilers and interpreters [HANS71a, TEIT79]. Many conversational languages such as BASIC, APL, and LISP include an editor that is actually considered part of the language environment [IVER62, BING76, KEME71, SAND78].

Interactive editors have been implemented on large time-shared computer systems, on dedicated microcomputers, and on word-processing systems designed primarily for office applications [CARL78]. On some document preparation systems the editor itself is interactive, but many of the formatting commands take effect only at the time the final printed copy is generated [OSSA77]. While all of these configurations fall within the purview of our survey, batch editors such as PANVALET [PANS77], which are designed to log all changes and to preserve back-up versions of programs in the maintenance of large software systems, and key data entry systems for massive data-processing operations [GILB77], do not.

To summarize, the functions of interactive editors of primary interest to us are (1) the creation, modification, and execution of computer programs; (2) the preparation of documents for human use; and (3) the ex-

amination and retrieval of portions of program, text, and data files.

Editor Design and Evaluation

In discussing editors firm opinions abound: everyone, from greenhorn to old hand, knows exactly what the best and worst features of given editors are and just how new editors ought to be designed. The only problem is the striking lack of consensus. Nor are there universally acceptable means of determining who is right: the distinction between conventional dogma and scientific fact is often blurred. There are, nevertheless, some established means of studying editors and the editing process. In the next few paragraphs we sketch the available sources of knowledge, starting with the most subjective.

Introspection, our own intuition and experience, is what we depend on when we assume that we know as much about the topic as the next person and are too lazy to look further. It is surprising how many programs intended for use by others, including text editors, appear to be based on this slim foundation, if we can judge by the lack of references to previous work in the publications describing them.

Field studies or *field observations* collect information *in situ* without substantial interference with the system or phenomenon under study. They range in scope from anecdotal evidence, which extends one's own experience to the isolated and fortuitous observation of others, to carefully recorded systematic observations carried out with deliberate planning of the features to be noted.

Formal analysis draws conclusions from a theoretical (for instance, syntactic or probabilistic) model of the editing system under study. Such analyses may suggest behavioral parameters for experimentation but have proved to be of limited value to date.

Controlled experiments restrict the number of variables to be manipulated and observed and attempt to minimize the effects of all other factors. The *observed* (or "dependent") *variables* are recorded as the *controlled* ("independent") *variables* are either held constant or driven, singly or jointly, through a predetermined range of interest.

Psychological models characterize human performance and add the important element of prediction. The principal goal of these models is to predict human behavior in a restricted environment while performing a set of tasks. Purely descriptive abstractions, on the other hand, cannot be extended to situations other than those for which all parameters are already known. Since behavioral studies that focus specifically on text editor usage are relatively scarce, we also draw on relevant studies performed in other contexts.

Applicable Areas of Psychology

The publication in 1971 of Gerald Weinberg's influential *The Psychology of Computer Programming* may be considered as marking the beginning of the behavioral approach to computer science [WEIN71]. As the continuing increase in the cost-effectiveness of computer equipment exposes more and more people without specialized training (and without the tolerance that reflects such training) to computers, many concepts and methods that are second nature to psychologists will find increasing application to the study of the human element [MILL77a, SHNE80].

As computer scientists interested in the study of text editors, we need not, fortunately, concern ourselves with all aspects of psychology. The area of psychology that appears most relevant to our needs is *cognitive psychology*, the study of higher mental processes such as memory, perception, learning, thinking, reasoning, language, and understanding. And as important as the topics themselves is the commitment to the observational view of science rather than a literary, intuitive, or humanistic point of view. An excellent introduction to the paradigms of modern cognitive psychology and a summary of the principal findings are presented in LACH79.

Bridging conventional pscyhology and the hard-science disciplines is the field of human factors engineering, or ergonomics, which burgeoned during World War II to deal with wartime problems in the area of skilled performance [WELF76]. Many inter-

esting examples of human-factors-oriented design can be found in DREY55. To quote from Lachman, the most important ideas borrowed by current research in information-processing psychology from human factors engineering are

(1) the view of man as an information transmitter and decision maker,
(2) the idea that there are limits to how much information he can transmit,
(3) the theory of signal detectability,
(4) continuing access to the concepts of the physical sciences,
(5) a reliance on sophisticated instrumentation, and
(6) a taste for federal funding.

While cognitive psychology abounds in elegant and sophisticated studies of mental processes, and many important facts have been established beyond dispute, its tools have been applied to the study of text editors only in rudimentary fashion. The weight given to various components of this survey is dictated by what we found in the literature.

Section 1 presents the development of temporal models, in which the duration of an editing task is predicted by analysis of its constituent components (e.g., command entry). The behavioral effects of the duration and variability of computer response times are also discussed.

The study of the impact of editor structure and command languages on behavior patterns is examined in Section 2. We review attempts to develop methods for differentiating among the characteristics of popular present-day editors ostensibly designed for similar applications.

Among the major achievements of cognitive psychology has been the development of a series of models for the accurate prediction of reaction times as a function of stimulus and response modality, number, complexity, and similarity. Reaction time is one of the favorite tools of experimental psychology. *Simple reaction time* (one stimulus, one response) is the time it takes to press a button when a light goes on (about 180 ms). Complex tasks, which include many stimuli and many responses (such as typing), require consideration of *stimulus categorization* and *response selection* in addition to simple reaction time. The discussion of key entry and pointing skills in Section 3 presents several attempts to extend classical reaction-time studies to more complex tasks.

The final section reflects our own opinions, based on this study, of where the most success has been achieved in applying cognitive psychology to text-editor design and evaluation, where the work appears to be headed for success, and where much greater effort and ingenuity are required.

1. PERFORMANCE TIME CONSIDERATIONS

An objective of the design and evaluation of text editors is to minimize the cost incurred by a user performing a number of editing tasks over a period of time. Ultimately, this cost is a function of the time taken by the user and the computer to complete each individual task. One may conjecture that task time depends on the nature of the task, the expertise of the user, the sluggishness of the machine response, and the time spent learning and relearning methods and procedures. Many factors, including the user's alertness and motivation, the availability of documentation and help, the editor's command structure, the ease of committing and correcting errors, and whether or not a hard-copy device runs out of paper, may also influence task time. Some of these time factors are beyond the control of the designer, but others can be evaluated and improved.

1.1 Simple Models for Predicting Task Time

Direct measurements of elapsed editing-session time are difficult to interpret because several not so easily controlled factors are introduced: the choice of editing commands, user alertness and motivation, and errors—both minor and disastrous.

Predictive models avoid these difficulties and further have the advantage of being useful at design time [CARD76, CARD78a, CARD80b, EMBL78]. These models are based on quantities such as keystroke count, typing rate, computer response time, and mental preparation time. The predictive power of these models depends on how accurately the constituent quantities can be estimated and the validity of any simplifying assumptions.

Card et al. propose a model that predicts task time for expert users performing routine tasks by simply accumulating the time required to perform individual unit tasks [CARD80b].

$$T_{task} = \sum_{\text{all units tasks}} T_{unit\ task}.$$

A large editing task, such as making numerous changes, additions, and deletions in a document, is considered to be a series of small, cognitively manageable, quasi-independent subtasks, called *unit tasks*. A unit task may correspond to a single command of an editor, or it may correspond to a short sequence of related commands to perform an action such as moving a block of code. It may also be a lengthy task such as typing in a document from a manuscript. Essentially, a unit task is a subtask for which a user has an available method and is describable in a simple phrase or two.

Unit task time depends on how long it takes a user to acquire a mental representation of the task and then perform or execute it,

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

The execution time is further refined as

$$T_{execute} = T_k + T_p + T_h + T_r + T_m$$

where

$T_k = n_k t_k$ is the *keying time* and depends on the number of keystrokes, n_k, in the unit task and on the typing rate, $1/t_k$, t_k = 0.2–1.0 second [SEIB72].

$T_p = n_p t_p$ is the *pointing time* and depends on the number of times the user points at something on the screen, n_p, and on the estimated time it takes to reference a screen position with a cursor under control of a "mouse," t_p = 1.10 seconds [ENGL67, CARD78b].

$T_h = n_h t_h$ is the *homing time* and depends on the number of times the user's hands move from one device to another, n_h, and on the estimated time for hand movement between any two devices, t_h = 0.40 second [CARD76, CARD78b].

T_r is the *response time*, which is command dependent, and is considered only if the user has to wait.

$T_m = n_m t_m$ is the *mental time* and depends on the number of decisions that must be made, n_m, and on the estimated time to make a decision, t_m = 1.35 seconds [CARD80b].

With the exception of mental time, it is easy to count the instances of each of these elemental actions and to accumulate the times, given the details of the methods used to accomplish the unit tasks. Mental time represents the time the user takes to prepare for executing a physical action. It is assumed to be constant and contributes to the task time whenever a decision must be made, except when the decision can be overlapped with such independent components as computer response time. Heuristic rules that specify how to count the contribution of mental time during a unit editing task are established and, in essence, assert that unless the next operation is anticipated by a previous one, a mental operation occurs [CARD80b].

As an example of how $T_{execute}$ is calculated, consider the task of replacing one word of arbitrary length with a five-letter word. The task can be performed as shown in Figure 1 using DISPED (an experimental display-based system at the Xerox Palo Alto Research Center). Assuming the user is an average skilled typist, the time per keystroke, t_k, is 0.20 second, and the predicted time to execute the unit task is 6.2 seconds.

An experiment to determine how accurately this keystroke model predicts performance times was conducted. Twelve subjects performed ten versions of four different editing tasks on each of three different editors—POET (a dialect of the QED Editor) [DEUT67], SOS [SAVI69], and DISPED—for a total of 480 observations. To avoid transfer effects, no subject was observed on more than one editor. All subjects were experts on the system they used, and all of the tasks performed were routine, ranging from a simple word substitution to the more difficult task of moving a sentence. Methods for accomplishing the four

Action	Constituents	Time (seconds)
Initially, the user's hands are on the keyboard		
1. Reach for mouse	t_h	0.40
2. Point to word on screen	t_p	1.10
3. Press button to "select" word	t_k	0.20
4. Home hands on keyboard	t_h	0.40
5. Execute replace command (type "R")	$t_m + t_k$	1.55
6. Type new five-character word	$5t_k$	1.00
7. Terminate entry	$t_m + t_k$	1.55
		6.20

FIGURE 1. Action sequence for replacing a word of arbitrary length with a five-letter word using the DISPED editor.

TABLE 1. CALCULATED AND OBSERVED EXECUTION TIMES[a]

Task	*System*	*Calculated* $n_k \times t_k + n_p \times 1.10 + n_h \times 0.40 + T_r + n_m \times 1.35 = T_{execute}$							*Observed* $T_{execute}$ (*mean*)	*Prediction Error* (%)
		n_k	t_k	n_p	n_h	T_r	n_m	$T_{execute}$		
1	POET	15	0.23				4	8.8	7.8	11
	SOS	19	0.22				4	9.6	9.6	1
	DISPED	8	0.23	1	2		2	6.4	5.7	11
2	POET	14	0.28				4	9.4	8.9	5
	SOS	18	0.23				4	9.5	9.7	−3
	DISPED	4	0.24	1	2		2	5.6	4.1	26
3	POET	12	0.19				3	6.3	6.3	0
	SOS	7	0.23				2	4.3	4.0	8
	DISPED	2	0.23	1	1		1	3.3	3.5	−7
4	POET	92	0.19				13	35.3	37.1	−6
	SOS	47	0.23				12	26.8	32.7	−22
	DISPED	6	0.24	3	1	3.8	2	11.6	14.3	−23

[a]Adapted from CARD80b.

different editing tasks were prescribed. Each experimental session lasted approximately 40 minutes and consisted of a practice session and a test session in which subjects performed editing tasks marked on manuscript pages in red ink.

Results are shown in Table 1. Tasks on which there were significant errors or in which the user did not use the prescribed method were excluded from consideration. As can be seen, predicted execution time matched the mean observed time reasonably well for most tasks but varied widely in a few instances. It must be remembered, however, that this is for a single unit task; because of the law of large numbers, predicted time for a succession of unit tasks should be more accurate.

In order to predict the total task time, task acquisition time was also estimated: 1.8 seconds to look at the manuscript only and 4.0 seconds to look at both the manuscript and screen. With this estimate of acquisition times, the prediction of task times by the keystroke model was accurate to within 5 percent.

Questions naturally arise as to whether simplified versions of this keystroke model might predict session time equally well or whether an even more detailed analysis would yield better results. Card and his colleagues analyzed several simplifications but found none to be as accurate [CARD80b].

A similar model for line-oriented editors had been proposed earlier by Embley and his colleagues, who describe a keystroke model in which acquisition time and mental time are considered as a single parameter and in which unit tasks are single command-response pairs [EMBL78]. Specifically,

$$T_{task} = mT_c + nT_k,$$

TABLE 2. OBSERVED DIFFERENCES BETWEEN NUROS AND CMS[a]

Program	Entry[c]	Normalized Task Time Difference[b]			
		Novice		Intermediate	
		Modification 1	Modification 2	Modification 1	Modification 2
1	10	−27	−9	18	19
2	19	−34	−8	19	17
3	14	11	−21	35	18
4	25	−18	−49	33	52
	17	−19.4		26.4	
Mean					
Confidence interval[d]	±8.5	±10.3		±9.0	
Conclusion	CMS better	NUROS better		CMS better	

[a] Adapted from EMBL78.
[b] Normalized task time difference is computed by

$$\frac{\text{NUROS task time} - \text{CMS task time}}{\text{NUROS task time}} \times 100$$

[c] The initial entry is identical for both Novice and Intermediate command sets.
[d] The confidence interval of the mean is calculated with Student's t-test at 95 percent certainty.

where m is the number of command–response pairs, T_c is the delay per command consisting of the mental preparation time and the computer response time, n is the number of keystrokes, and T_k is time per keystroke. The quantities m and n depend only on the editing task to be performed and on the available command language; thus if m and n can be measured with reasonable accuracy, then the duration of task time can be predicted for various values of command delay time and typing rate.

An objective of this study was to investigate a procedure for comparing program editor performance as a function of the time required for a user to perform editing tasks. By way of example, the model was applied to command subsets of NUROS [UNCN79] and CMS [IBM76] suitable for novice and intermediate programmers. Twelve editing tasks were defined by arbitrarily selecting three versions of four student programming projects—an initial version, an intermediate version, and a final version. Four sets of commands were established: the NUROS novice and intermediate sets and the CMS novice and intermediate sets. With these command sets an experimenter (an expert user of both CMS and NUROS) studied the editing tasks and then performed them. Command–response pairs (m) and keystrokes (n) were counted, and the model was applied with parameters $T_c = 5$ seconds and $T_k = 0.5$ second. The results are shown in Table 2.

Actual task time was not an objective of this study, so empirical data for comparison with the keystroke model of Card and colleagues are not available. Since the keystroke model makes a finer distinction of the delay per command, however, it should be a better predictor.

1.2 More Comprehensive Models

Models with very detailed mental-time operators have also been proposed. Treu [TREU75] presents a model for the mental work involved in text-editing tasks that is based on action primitives and their relationship to system commands, but reports no experimental data to verify his hypothesis. An experimental study where the question of level of detail is addressed was performed, however, by Card et al. in an investigation of their GOMS model of text editing [CARD76, CARD80a].

The GOMS models describe a user's performance in terms of *goals*, *operators*, *methods for achieving the goals*, and *selection rules for choosing among competing methods* (Figure 2). It is a theory that at-

GOMS Constituents	Examples
Goals	Edit manuscript Edit unit task Locate line Modify text
Operators	Use substitute command Look at manuscript Verify edit
Methods	For locating a line alternative methods are string search move forward/backward combinations of the above two methods For changing a line some alternatives are delete-insert change
Selection rules	For locating a line selection rules might be Rule 1: Use string search as default Rule 2: Use forward/backward if the distance is known For changing a line selection rules might be Rule 1: Use change command as default Rule 2: Use delete-insert if it takes fewer keystrokes

FIGURE 2. Sample goals, operators, methods, and selection rules for a GOMS model.

tempts to explain *how* an expert user accomplishes routine editing tasks and thus models more than just the various time constituents that comprise task completion time.

A feature of the GOMS model is its ability to adjust to either more or less detail. The substitute command, for instance, can be expressed as a unit or, in more detail, as "specify substitute command-specify argument number 1-specify argument number 2-enter command." The level of detail is called the *grain* of analysis. As the grain becomes finer, the models expose lower level operations and render them susceptible to measurement; thus one might argue that these fine-grained models are potentially more accurate. It is known, however, that the times required for operators in a sequence may be interdependent, especially in a fine-grain analysis [ABRU56]. Furthermore, measurements of individual fine-grain operators are typically less accurate than measurements of coarse-grain operators.

Since it is difficult to know what grain size to use, several variations of the GOMS model were explored. The models ranged from "very coarse," where a single operator of constant duration represents each unit task, to "quite fine," where operators such as "home hands on keyboard" or "type an s" are of less than half-second duration. Within this time stratification models on the same level differed by the degree that alternative operators (or sequences of operators) were considered. For example, one model with 4-second operations used single operators for each functional step ("locate line," "modify text") whereas another model at the same level considered the same functional operators but divided them into separate cases on the basis of the methods used to accomplish them ("string search" or "move forward/backward" rather than "locate line").

In order to test the GOMS model and to see the effects of the grain of analysis, an experiment was conducted using ten different GOMS models: one model at the level of about 16-second operator duration, two at about 4-second operator duration, four at about 2-second duration, and three at about 0.5-second duration. Although five subjects participated, the volume and detail of data permitted an intensive analysis of

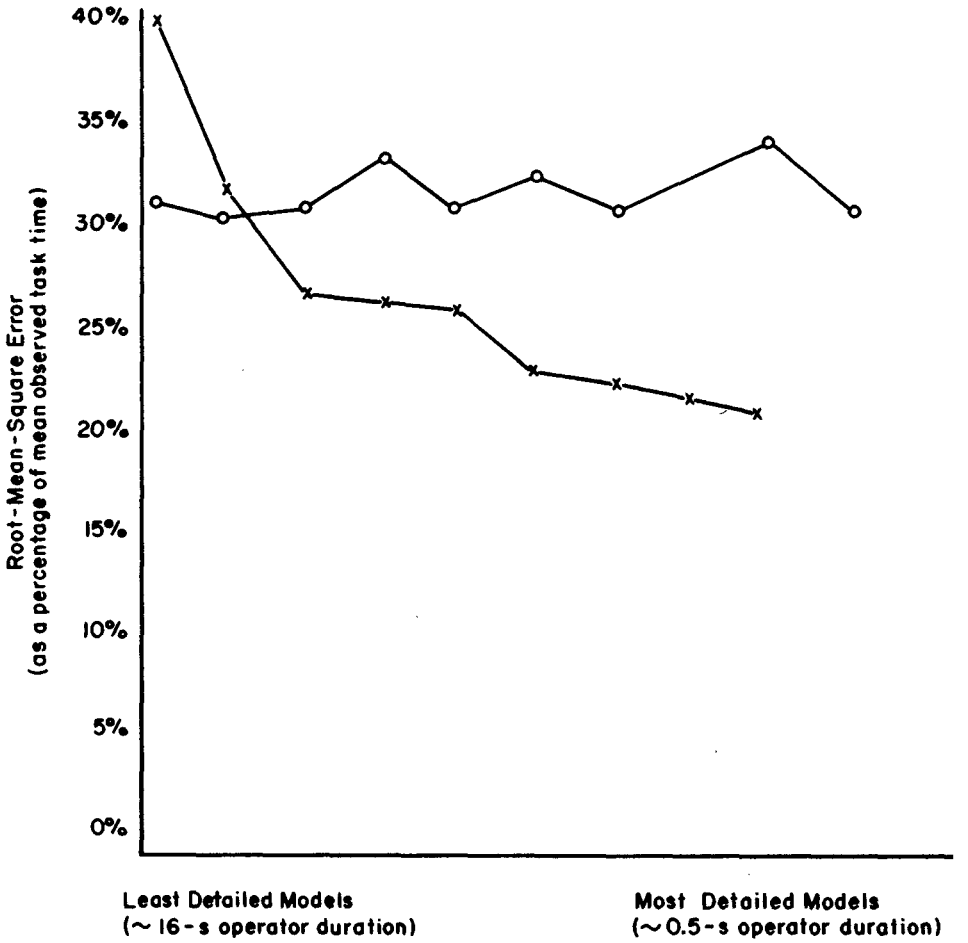

FIGURE 3. Accuracy of GOMS models (adapted from CARD78a). x, reproduction of derivation data; O, prediction of cross-validation data.

only one subject. The data from this subject were partitioned into two sets, a *derivation* data set from which prediction rules for operator sequences and estimates for operator duration were derived and a *cross-validation* data set which was preserved for calculation of unit task duration using predicted operator sequences and durations. To obtain an upper bound on the models' predictive power, the duration of unit tasks in the derivation data were also calculated using the actual operator sequences.

The results are shown in Figure 3. The root-mean-square difference between a set of predicted and observed unit task times is expressed as a percentage of the average observed unit task time. The accuracy of the reproduction of the derivation data improves from just under 40 percent when the average unit task time is the predictor to about 20 percent for the most detailed model. The main result, however, is that the predictions based on the cross-validation data are all about equally accurate, with a root-mean-square error of about 30 percent of the mean observed task time. Grain size of the model appears to be of little consequence unless the sequence of operators can be predicted nearly perfectly.

Although an error of 30 percent seems high, predicting editing times unit task by unit task for a single user is a very stringent test. If task time rather than unit task time were predicted using a predictor statistically adjusted to be unbiased, then the percent prediction error would drop approxi-

TABLE 3. SELECTION RULES FOR THE LOCATE GOAL[a]

User	Rule	Percent of Cases Predicted Correctly (Cumulative)
S1 (tty)	1. Search by specifying a string in the desired line unless another rule applies	65
	2. If the number of lines to the next text to be modified is less than three, move forward/backward one line at a time	78
	3. If the desired line is the last line of a page, search by specifying the end of page marker (a $)	85
S2 (tty)	1. Move forward/backward by specifying number of lines to move unless another rule applies	77
	2. If the number of lines to the next text to be modified is fewer than three, move forward/backward one line at a time	94
S2 (CRT)	1. Move forward/backward one line at a time unless another rule applies	68
	2. If the number of lines to the next text to be modified is greater than nine, move forward/backward by specifying the number of lines to move	74
	3. If the desired line is on the next page of the manuscript, move forward one line at a time	85
S3 (CRT)	1. Search by specifying a string in the desired line unless another rule applies	62
	2. If the number of lines to the next text to be modified is fewer than five, move forward/backward one line at a time	92
Average predicted correctly by use of all rules for each subject		89

[a] Adapted from CARD80a.

mately as the square root of the number of unit tasks. For an editing session consisting of about as many (73) unit tasks as were marked on the manuscript, the prediction models would be accurate to within 3–4 percent.

Besides the investigation of time prediction and analysis of grain size, Card and his colleagues also investigated how accurately the GOMS model might predict the methods a user would select to accomplish a task. The objective was to determine whether a set of simple selection rules could account for the methods users select.

An experiment was conducted in which the method selection to locate a line with the POET editor was observed. Four experiments were run in which subjects were given manuscripts with 73 corrections marked in red ink. In two of the experiments subjects simply located the line; in the other two experiments subjects also edited the manuscript. Two of the experiments were run with teletypewriters and two with CRT displays. Three subjects participated; one subject repeated the experiment after a two-week interval and performed one experiment on a teletypewriter and the other on a CRT.

A summary of the results is shown in Table 3. Each subject appeared to have a dominant method—the rule listed first. Apparently users apply the dominant method unless it is obviously inefficient. Note that S2 applied one dominant method while using the teletypewriter and another dominant method while using the CRT, presumably because of the speed difference between devices. The selection of methods also depends on the features of the task. For locating a line the most important characteristic of the task is the number of lines between the current line and the line with the text to be next modified.

Observing (1) that a high percentage of the methods selected can be accounted for by a few simple rules and (2) that expert users certainly do not take time to make elaborate calculations to determine which method to use leads to the conclusion that users are able to quickly select near-optimum methods by having assimilated heuristic rules based on a few pertinent task features. Since it might be conjectured that if users cannot easily choose among alternatives, they will either ignore one of the methods or will agonize over which to use, designers who provide alternative methods

TABLE 4. AVERAGE PERFORMANCE TIMES AND VARIATIONS[a]

	Mean Time in Minutes to Perform Task				*Coefficient of Variation (CV = Standard Deviation/Mean)*	
	TECO	*Wylbur*	NLS	*Wang*	CV *of Means*	*Mean of* CV
Total-time data	47	42	30	24	0.30	0.28
Error-free data	41	35	24	22	0.30	0.18

[a] Adapted from ROBE79.

for accomplishing goals should have in mind clear decision rules for deciding among alternatives.

1.3 Observations of Overall Task Time

In the studies of the keystroke and GOMS models, it is assumed that users are experts and that they perform editing tasks perfectly—without error. For the data analyses error data were either discarded or folded into the data as if no errors had occurred. Even for expert users, however, from 5 to 30 percent of actual editing time can usually be attributed to errors and error corrections. If an accurate prediction of task time is desired, errors must also be considered.

In an attempt to provide a comprehensive basis for the evaluation of text editors, Roberts investigated errors and also considered text editors from the point of view of several kinds of users doing different kinds of work [ROBE79]. Because of practical limitations, however, Roberts actually developed and performed only a few of the many experiments she suggested.

In one of the experiments she investigated task time on four text editors: TECO [BOLT73], Wylbur [STAN75], NLS [AUGM75], and Wang [WANG78]. Four experts performed four separate editing tasks: they entered a short memorandum, modified two business letters, and corrected an excerpt from a text on philosophy. Since one of the objectives of the study was to provide evaluation schemes that are quick and straightforward to apply, she did not assume the availability of sophisticated data-recording equipment. Instead, a human observer noted the time at the beginning and end of the task and used a stopwatch to obtain the time spent making and correcting errors. Only errors that took more than 30 seconds to correct were recorded; small mistakes such as typographic errors that were caught and corrected immediately were folded into the editing time. The observer also kept track of tasks accomplished incorrectly or skipped, and directed subjects to correct or complete them. This time was added to the error time.

A summary of the results appears in Table 4 and shows that task times were considerably longer with TECO and Wylbur than with NLS and Wang (with statistical significance at the 0.02 level). The coefficients of variation for the total-time data indicate that differences between subjects account for about as much variability as differences between editors. For the error-free data, however, more variation can be attributed to the editors than to the subjects. Thus much of the subject-to-subject variation must be due to error rates.

Roberts also applied the keystroke model of Card et al. to her data to predict task times. The editing sequence chosen for the model was a sequence of optimal methods for each subtask in its context. Since some subtasks were larger than unit tasks as defined for the keystroke model, each get-locate-modify-verify cycle was considered a new unit task.

Results are shown in Figure 4. Ignoring the error data—since the keystroke model assumes that the data are error-free—the bar graph shows that the pedictions were 25–50 percent too low, and therefore not nearly as accurate as expected in light of the validation experiments performed on the keystroke model itself. Further investigation of the data obtained on TECO (an automatic record of all keystrokes was kept) showed that application of the model to the actual keystroke sequence still accounted only for 87 percent of the error-

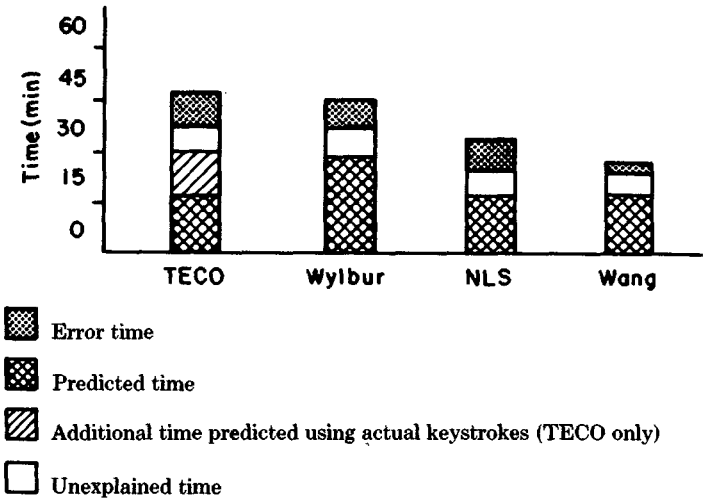

FIGURE 4. Results and accuracy of task-time predictions [ROBE79].

free time. The remaining time was attributed to unknown mental activities.

To explain the discrepancy, it must be remembered that in the keystroke-model-validation experiments, methods for each unit task were suggested and practiced before the experiment, task acquisition was nominally excluded, and error data were completely eliminated, whereas in Robert's data small errors were included in the reported times. Moreover, prediction is more difficult when users choose their own (possibly suboptimal) editing sequences. The keystroke model predictions provide, however, an upper bound on how well a skilled user could perform if he or she were so practiced that method selection time would be nil, choices optimal, and entry flawless. A comparison of predicted time and observed time therefore provides useful information since a relatively large difference indicates that the editor under consideration is difficult to use optimally.

1.4 The Effects of Computer Response Time

The importance of system response time is universally recognized. Since the earliest days of interactive computing, researchers have discussed the effects of system delay and unpredictability on user productivity and satisfaction [CARB68]. Although some controlled experiments have been conducted and have produced interesting and worthwhile results, they have addressed the broader issues of interactive computing such as information retrieval, interactive design, and problem solving rather than text editing [BOEH71, GOOD78, GROS76, MILL77b, YULE72]. For all of these interactive activities, mental preparation is more intense and varied than for routine editing tasks and is thus more sensitive to interruption, distraction, and unusual delays. Editing should proceed at a rapid pace; for most requests any perceptible delay may prove irritating. For a good overview of system response time issues in the broader context of interactive computing, see Shneiderman [SHNE79].

R. B. Miller maintains that an immediate response is not a universal requirement in interactive computing and lists various classes of user actions and purposes at terminals that appear to allow or require different system delays [MILR68]. "Goals," "chunks," and "closures" all play an important role in determining acceptable delay. Classifications that apply to editing include echo characteristics, "conversational" requests, searches, task completion, log-on/log-off, and recovery from system failure (Figure 5). It is emphasized that these estimates are "best guess" conjectures of a behavioral scientist who specializes in computer usage; he urges that they be verified by extensive system studies in carefully designed, real-life task environments.

Some results from controlled experiments on the effects of system response

Classification	Acceptable Time Delay (Seconds)	Comments
Echo characteristics	<0.1	Examples are the click of a typewriter key and visual feedback from the platen or CRT
Conversational requests	<2	Most editing requests fall into this category
Searches		
String search requests	<4	Two-second delays would be preferable, but if the user perceives the request as a complex inquiry, up to 4 seconds is acceptable
Browsing, page-by-page search	<0.5	Longer delays, at least for the appearance of first several lines of the next page, are intrusive on the continuity of thought
Task completion		
Program execution	<5	There is a sense of closure, but if continuity is desired, the delay should not be too long
New editing assignment	10–15	A brief rest is nice
Log-on/log-off	<15	Captivity of more than 15 seconds can easily demoralize the user and reduce motivation to work
Recovery from system failure	<15	If recovery will take more than 15 seconds, the system should inform the user how long he/she might have to wait

FIGURE 5. Subset of Miller's response-time classification that is applicable to text editors.

time in interactive systems are applicable to editing, but care must be exercised in transferring results from one environment to another. L. H. Miller investigated the effects of varying CRT display rates and output delays on user performance and attitudes in a series of message retrieval tasks [MILL77b]. He concluded that increasing the display rate from 1200 to 2400 baud produced no significant performance or attitude changes, but that increasing the variability of the output display rate produced a significant deterioration in both performance and attitude. It should be noted that 1200 baud is much faster than the average person's reading rate, which is closer to about 300 baud (approximately 360 words per minute); presumably too slow a communications rate would have a deleterious effect. In the editing environment, however, there could be a significant difference between 1200 and 2400 baud for some situations—for instance, when skimming in search of an item to be modified. (The typical communication rates for terminals are rising gradually—110-baud teletypewriter connections common a few years ago have been largely replaced by 300-, 2400-, and 4800-baud dial-up lines. Most standalone computers have 9600-baud screen display rates.) The effects of variability, as shown in the experiment, are also likely to be detrimental in the editing environment; such variability is not uncommon with line-by-line transmission.

Grignetti and Miller conducted experiments to explore methods to motivate users to adopt behavior patterns that would improve overall system performance [GRIG70]. For example, text modification could be performed with either a machine-cycle intensive search-and-replace command or a keystroke intensive delete-and-enter sequence. They investigated both controlled computer response time and an imposed cost-reward structure to regulate user requests. Their experiments demonstrate that it is possible to provide incentives that affect choices between alternative methods of accomplishing a task. They discovered, however, that even with very explicit monetary incentives users do not make optimal choices. Instead, the strategies appear to be based on some perceived

cost that is related but not identical to the actual cost.

In another experiment Grossberg and his colleagues studied response times in problem-solving activities with mean delays of 1, 4, 16, and 64 seconds but with individual delays varying widely and unpredictably about the mean [GROS76]. In harmony with the findings of Grignetti and Miller, results show that subjects modified their problem-solving tactics as the mean delay increased—they seemed to become more cautious and deliberate. Surprisingly, however, the mean delay did *not* have a definite effect on the time required to reach a solution. In transferring the results to the editing environment, one might conclude that system response time has little effect on performance; users would simply adjust their tactics to make the best use of their time on the system. A major difference between a problem-solving and an editing environment, however, is the usefulness of extra mental time during long system delays. Since editing is typically a routine cognitive skill, additional mental preparation time beyond what is necessary to decide what to do next would likely interfere with the task completion rate. Moreover, since the subjects in this experiment were the experimenters themselves, they were motivated to complete the assigned tasks successfully, but this does not imply that other users would tolerate such abuse.

2. EDITOR DESIGN CONSIDERATIONS

In this section we consider aspects of editors that affect ease of use. Ease-of-use considerations are particularly important for novice and casual users, but also affect experts who wish to invoke infrequently used editor features or who use several different editors on a regular basis. Most authors agree that ease of use depends primarily on the command language and underlying structure of the editor and may also depend on the nature and availability of user aids. Relevant command language features may include the number of commands, the command vocabulary, mnemonics, abbreviations, arguments, defaults, and macros. The structure of an editor, though not independent of the command language, can be described in terms of editor states or modes, context dependencies, and state transitions. In essence, design trade-offs balance the proliferation of "powerful" commands that depend heavily on the editor's state against a small number of "basic" commands executed from a minimal number of states. Our inability to learn, remember, and effectively use large, complex command sets, balanced against our desire to achieve editing objectives with a minimum expenditure of effort and time, limits the range of reasonable design options.

In this section we survey four approaches to editor design and evaluation: popular wisdom, observation, analysis, and controlled experiments. Popular wisdom stems mainly from introspection and is typically influenced by anecdotal evidence. Observation encompasses field observations and surveys taken from users. Formal analyses address the issues of syntax and semantics of command language grammars. A few controlled experiments on editor learnability and user friendliness have been performed and yield valuable insights into editor design.

2.1 Popular Wisdom

The literature extols the virtues of many text editors [COUL76, DEUT67, HAZE80, TEIW79, VAND71] and is replete with lists of suggestions on how to create a better human-oriented interface (see, for example, GAIN78, JONE78, HANS71a, MART73, ROUS73, WASS73). Several of these lists are also compiled in SHNE79. As a single representative example of these design guidelines, Hansen's "User Engineering Principles" are shown in Figure 6.

Hansen expounds upon the meaning of each of these principles. For example, for *predictable behavior*, he explains [HANS71a]:

> The importance of such behavior is that the user can gain an "impression" of the system and understand its behavior in terms of that impression. Thus by remembering a few characteristics and a few exceptions, the user can work out for himself the details of any individual operation. In other words, the system ought to have a "Gestalt" or "personality" around which the user can organize his perception of the system.

These principles were developed during the design of Emily [HANS71b], a sophisticated program editing system for PL/1 in which text is created, viewed, and modified in terms of the structure imposed by the syntax of the programming language. (We note that at a later time and in a different setting Hansen did run a controlled experiment that tested in part the notion of predictable behavior [HANS78].)

Typically, guidelines such as Hansen's contain reasonable advice and guidance but are often vague and sometimes even contradictory. They suggest little more than what a good designer knows from experience and common sense and do not lead to quantitative methods of evaluating editors.

Martin [MART73] and Engel and Granda [ENGE75] present much more comprehensive guidelines. Martin discusses the user-computer interface, taking into account various hardware configurations, user abilities and objectives, and implementation considerations. Engel and Granda consider seven general categories; some of these recommendations are presented in Section 3.2. These guidelines were based on a thorough survey of information available at the time they were written and represent the assimilation of experience, informal observations, behavioral experiments, and principles thought to be applicable.

Another approach to the dissemination of popular wisdom is taken by Singer et al. [SING77, LEDG81]. They present an annotated user's guide for an editor, the PASCAL Assistant, with the intent to illuminate the human engineering design considerations and to explain the principles motivating their decisions. Figure 7 provides an example of comments about an aspect of the editor. Although they make a conscious effort to rely on psychological principles where possible, they freely admit that often their only guide was intuition and experience.

User Engineering Principles

First principle: Know the user

Minimize memorization
- Selection not entry (this agrees with FIEL78)
- Names not numbers
- Predictable behavior
- Access to system information

Optimize operations
- Rapid execution of common operations
- Display inertia (the display should change as little as possible to carry out a request)
- Muscle memory (subconscious muscle memory should be exploited, for instance, to maintain maximum keying rate)
- Reorganize command parameters (keep frequent commands simple; infrequent commands can be more complex)

Engineer for errors
- Good error messages
- Engineer out the common errors (the design may need to be altered to inhibit frequent user errors)
- Reversible actions (it should be possible to restore the system to a previous state)
- Redundancy (provide more than one means to an end)
- Data structure integrity (data should not be lost regardless of system or hardware malfunction)

FIGURE 6. Hansen's table of user engineering principles (adapted from HANS71a).

2.2 Observation

Opinions about the user-perceived quality of text editors abound, but actual knowledge is scarce. A long-range objective is to obtain a criterion to measure quality from the user's point of view, but a first step is to identify relevant system properties and user abilities and learn how to obtain data about them [EMBL81a].

One possible approach to identifying relevant properties and abilities is through questionnaires [DZID78]. On the basis of a pilot study, Dzida and his colleagues assumed that user-perceived quality could be seen as a multidimensional concept with each dimension representing an independent characteristic of the overall quality. They searched for these quality characteristics by means of questionnaires and statistical analysis of the results. First, 300 experienced users were asked to state relevant human-oriented system requirements; later, about 600 persons were asked to judge 100 system requirements with respect to their importance in user-perceived quality. Questionnaires were returned by 233 persons, about half of whom were among those who initially stated the relevant human-oriented system requirements and half of whom were members of the German chap-

User's Guide	Rationale
Active Behavior: When you send the Assistant a request, it becomes active and attempts to satisfy your request. It does this in three stages*: 1. Verification—The Assistant determines whether or not your request makes sense, and makes any necessary assumptions that it can when specific details are not given. 2. Performance—If the verification stage was completed successfully, the Assistant will satisfy your request. If the operation requested is at all time-consuming, the Assistant may indicate its progress at various intervals. 3. Completion—After your request has been satisfied, the Assistant indicates the final result of its actions and again becomes attentive.	* There is a body of psychological evidence (see, for example, Thorndike and Rock, 1934) which suggests that people "learn without awareness." One implication of these results is that the users of a computer system will infer underlying principles even if they are unaware of doing so. The Assistant's behavioral goals are not merely "sugaring," but are accurately reflected in its responses. These goals are intended to help the user make reasonable inferences about what the Assistant will do with a particular request. For example, the first goal, verification, ensures that no request will be executed unless it makes some sense semantically. In some cases, this implies that significant static prechecking must be performed. This seems a small price to pay for relieving the user of the burden of correcting damage done by a technically legal but senseless request.

FIGURE 7. Excerpt from the "The Annotated Assistant: A Step Towards Human Engineering" [SING77].

ter of ACM who had not previously participated.

Seven categories that accounted for 44 percent of the variation in the data were identified. These categories were denoted *self-descriptiveness, user control, ease of learning, problem-adequate usability* (minimize details the user must know and deal with), *correspondence with user expectations, flexibility in task handling,* and *fault tolerance.* These factors were isolated mathematically and at least five of them were shown to be statistically reliable and valid. The importance of some factors, however, varied widely depending on specific user groups. For instance, casual users felt, much more so than did regular users, that ease of learning was important. Since the seven-factor solution explained 44 percent of the variation in the data, the authors concluded that an empirical model for assessing user-perceived quality had been established.

Another approach to identifying relevant characteristics is through direct observation of users on existing systems. Kennedy observed a large sample of clerical and secretarial staff learning to use an interactive system and gained insight into the effect of anxiety and the role of the system, the instructor, and reference manuals in the learning process [KENN75]. Initially, this field observation began as a controlled experiment to investigate factors that might affect learning, such as attitude toward computers, availability of manuals, self-learning from the system, and verbal assistance from an instructor. In the experiment none of these factors was shown to be significant, but observations did lead to system improvement and more effective training. For the particular interactive system, Kennedy observed that (1) self-teaching through trial and error with feedback from the machine seemed most effective, (2) subtle distinctions in technical terminology were inadequately explained, and (3) anxiety decreased learning, particularly during the subject's first computer session. Observations of this nature yield new hypotheses that can be tested. Even when experiments do not produce expected results, they may provide useful insights and experience.

Users may also be observed indirectly by instrumenting editors to extract and time-stamp editing sessions. Hammer and Rouse collected the sequence of keystrokes and the elapsed time between keystrokes for researchers writing their own programs and reports using TECO and SOS [HAMM 79]. The data collected were mapped into sequences of uniform editor primitives (e.g.,

insert one line, delete many characters) in order to make a comparison between the data from the two editors possible. As an indicator of the editing sequence, digrams of uniform editing primitives were counted, and the counts were converted into a transition probability resulting in a Markov model. Statistical tests applied to the Markov model showed that differences between editors and between tasks were no larger than the differences between users. For 22 of the users involved in both program and document editing, statistical tests showed significant differences between tasks for only 25 percent of the users; most of the observed users edited programs and documents by the same technique. Hammer and Rouse have begun to apply their Markov model to higher level editing operations in an attempt to capture more task-specific behavior and to separate task differences from individual differences. They also plan to study errors and the delay between keystrokes.

Sample Predictions

1. Learning and/or remembering how to select shapes in ROBART 1 should vary in difficulty.
2. Learning and/or remembering how to select shapes in ROBART 2 should *not* vary in difficulty.
3. Learning and/or remembering how to select any shape in ROBART 2 should be easier than selecting the corresponding ROBART 1 shape.

Results Related to These Predictions

Number of subjects (of ten) unable to select the given shape:

Task	ROBART 1	ROBART 2
Line	0	0
Box	4	1
Circle	8	0
Continuous line	2	0
Continuous box	6	0
Continuous circle	9	1

FIGURE 8. Some predictions from an analysis of the grammars of ROBART 1 and ROBART 2, and results from the exploratory experiment conducted (adapted from REIS81, ©IEEE 1981). (ROBART 1 and ROBART 2 are two versions of an interactive graphics system for creating slides that have essentially the same function but differ in the design of the human interface.)

2.3 Syntactic Analysis

Besides user feedback through questionnaires and direct observation, another way to gain insight into relevant characteristics is through an analysis of formal descriptions of command language grammars [LEDG78]. Although formal grammatical descriptions have not generally been applied to study the subjective aspects of editors, Reisner [REIS79] and Anandan [ANAN79] illustrate through examples how formal descriptions can be used in human factors research. Reisner describes user actions at a terminal for two command languages by means of a BNF-like grammar. Aspects of the formalism Reisner considered to be useful for comparison include the number of different terminal symbols, the lengths of the terminal strings, and the number of rules necessary to describe the structure of some set of terminal strings for a given task. An examination of these aspects of the formalism led to predictions about user behavior. In order to test these predictions, an exploratory experiment was conducted in which subjects learned both command languages and performed tasks designed to reveal information about the predictions. Reisner observed that subjects performed consistently with the predictions (Figure 8).

In a similar vein, Anandan developed state transition diagrams for two editors, NUROS [UNCN79] and SIMPLE [EMBL81b], and counted the number of states, number of different commands, number of commands issued from each state, total number of context dependencies, and average number of keystrokes necessary per command (Table 5). The data reveal differences between the editors and correspond with informal observations of the differences in ease of use.

Several different descriptive notations appear useful in this approach to assess user-perceived quality, and it is not clear which is best. Besides those mentioned here, Moran's command language grammar [MORA81] looks promising since it describes all levels of a command language system, from the conceptual to the physical device level.

Beyond the choice of formalism, however, lies the more difficult problem of attaching measures of user effort to the quantities that can be extracted from the formalism. It seems relatively easy to single

TABLE 5. STATE TRANSITION DIAGRAM SUMMARIES FOR NUROS AND SIMPLE[a]

NUROS		*SIMPLE*	
States	*Number of Commands*	*States*	*Number of Commands*
1. Begin mode	6	1. Command selection mode	11
2. Ready-to-write mode	2	2. Text insertion mode	2
3. JCL mode	3		
4. File-display-write mode	9		
5. Edit mode	5		
Total number of commands issued from states	25		13
Number of distinct commands	17		13
Number of commands issued from more than one state	7[b]		0
Average number of necessary keystrokes	3.9		2.5

[a] Adapted from ANAN79.
[b] Six from two different states and one from three different states.

out a particular factor, collect data from two comparison languages, apply some measure, compare the results, and declare that one feature is better than another, but it is difficult to determine *how much* better and *how much weight* the factor should have. Moreover, when complex interactions among several factors are considered, the results might not turn out as expected. Perhaps some ideas adapted from software science studies [HALS77] may provide clues to the answers to these questions.

2.4 Controlled Experiments

Since very few controlled experiments have been conducted to assess the quality of text-editor command languages and system structure, it is necessary to draw information from behavioral experiments that bear on some aspect of text editors. One must be careful when transferring the results of a behavioral experiment from one context to another—the assumptions may not hold and the context may be sufficiently different to invalidate the results completely. With this caution in mind we examine a few examples.

Freedman and Landauer [FREE66] and Chin-Chance [CHIN78] have obtained data that indicate the usefulness of the initial letter of a word as a recall and discrimination clue. Permitting first-letter abbreviations for command names gains support from these data.

Newman investigated imperative statements with preconditions of two logical types, "standing" and "one-shot" [NEWM77]. Standing preconditions are assumptions that normally hold true in a given situation. For example, the statement "if you wish to edit file F, issue the command 'open F'" has a standing precondition, "if you wish to edit file F." One-shot preconditions are exceptions or are events that occur only once during the accomplishment of a task. For example, the statement "if you wish to edit file F but it doesn't exist, issue the command 'create F'" has a one-shot precondition indicated by the phrase "but it doesn't exist." The results tend to support the hypothesis that the semantic processing of one-shot commands is more complex (because more conditions must be processed). Thus, for creating a new file, for example, Newman suggests that it may be preferable to design an editor with only a single command to open and create a file.

A few controlled experiments have been conducted that specifically address command language structure and learnability aspects of text editors [ROBE79, WALT74, LEDG80]. Walther and O'Neil investigated interface flexibility; that is, whether user options are good for everyone's performance and, if not, for which kinds of users they are helpful (or detrimental). Two different versions of a text editor were specif-

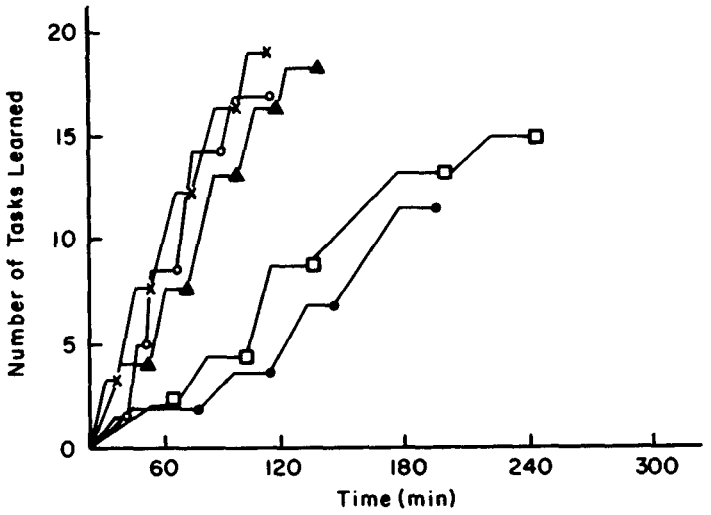

FIGURE 9. Average learning curves [ROBE79]. □, TECO; ●, TECO (second instructor); ○, Wylbur; ▲, NLS; x, Wang.

ically designed and constructed for the experiment. The inflexible version required all commands to be spelled out fully: no abbreviations, extra spaces, or defaults were allowed. The flexible version permitted as much freedom as possible so long as ambiguities could be resolved. Measures of user performance were derived from time-for-task and syntax-error-frequency data. The somewhat surprising results indicate that interface flexibility is not uniformly best for all users. In the experiment users were more prone to make syntax errors when offered more flexibility, but completed tasks faster than those not having the options. The exceptions were the novices who worked more rapidly without options than with them. Unfortunately Walther and O'Neil's paper does not include their quantitative results [WALT74].

Roberts investigated the question of how long it takes a beginner to acquire the ability to perform basic editing operations—insertion, deletion, replacement, moving, copying, splitting, and merging [ROBE79]. Under the guidance of an instructor, sixteen subjects, four for each of the four text editors (TECO, Wylbur, NLS, and Wang), learned to perform these basic tasks. A task-oriented instructional method was standardized across all editors; the instructor taught and quizzed subjects on the techniques and commands needed to perform five different teach-followed-by-quiz cycles covering the basic tasks. Time spent in both being taught and quizzed was controlled by the subject, and the results are presented as a learning curve with the number of tasks the subject had shown the ability to perform plotted against time (Figure 9). Since the learning rate undoubtedly depends strongly on the particular instructor, a second instructor ran the experiment on four additional subjects using the TECO text editor. A one-way analysis of variance of the data shows that there is a significant difference ($p < .01$) among systems. The t-tests between systems show a significant difference between TECO and each of the other systems ($p < .05$), but no difference among the other three systems or between the two sets of TECO experiments.

Roberts also explored the possibility of analytically predicting the results of these experiments in an attempt to find a less expensive means of evaluating alternative systems and to gain some insight into which features of editors affect the learning rate. She counted the commands (n1) and items (n2—the entities such as verbs, arguments, and terminators that comprise commands) necessary to learn the basic operations for each of the four editors. Table 6 shows the results of these counts plus average counts

TABLE 6. PREDICTION OF LEARNING RATES[a]

	TECO	*Wylbur*	NLS	*Wang*	*Standard Error*[b] (min/task)
Observed learning rate (minutes per task)	19.5	8.2	7.7	6.2	
Predicted learning rate from command vocabulary (n1)	17.4 (14)[c]	4.9 (29)	9.9 (11)	9.9 (11)	4.1
Predicted learning rate from item vocabulary (n2)	12.4 (34)	9.9 (29)	13.9 (37)	7.9 (25)	6.8
Predicted learning rate from commands per task (m1)	19.4 (4.0)	7.5 (1.3)	6.6 (1.1)	8.4 (1.5)	1.8
Predicted learning rate from items per task (m2)	17.5 (11.5)	11.2 (8.0)	7.4 (5.9)	4.5 (4.3)	2.8

[a] Adapted from ROBE79.
[b] $[\Sigma (t_p - t_o)^2/(N-2)]^{1/2}$ where t_p is the predicted rate, t_o is the observed rate, and N (=4) is the number of trials.
[c] The numbers in parentheses represent the command and item counts used to predict each learning rate.

of commands per task (m1) and items per task (m2) and the result of a regression analysis.

It is interesting to note that of these counting methods, command counts seem to predict learning rates better than item counts, a somewhat counterintuitive result since the command count is a cruder measure. Furthermore, it would appear that it is not the absolute number of command (or item) types that matters, but the number of commands (or items) that must be strung together to perform an elementary editing task. Since these results are based on only four sample points, the results are far from definitive, but the hope remains that analytical predictions of this nature may someday be regarded as valid measures for assessing an editor's strengths and weaknesses with respect to learnability.

Ledgard et al. hypothesized that an interactive system should be based on familiar, descriptive, everyday words and English-like phrases [LEDG80]. They observe that this hypothesis is not generally accepted in the design of the vast majority of interactive languages. To test their hypothesis, they compared two text editors: a commercially available Control Data Corporation editor supplied with NOS that has a typical notational syntax, and a remodeled version of this editor with identical power but with its syntax altered so that its commands are all based on legitimate English phrases. Twenty-four paid subjects, distributed among three levels of familiarity (eight novice, eight intermediate, and eight expert), participated in the experiment. Each subject performed editing tasks on both versions of the editor; half of the subjects at each level of familiarity tried the notational version first and half tried the English-based version first.

The results are summarized in Table 7. Subjects completed more tasks with the English-based editor ($p < .001$). The error rate for the English-based editor was lower ($p < .01$). The editing efficiency for the English-based editor was better ($p < .01$). Since both editors were semantically identical and since the performance on the English editor was strikingly superior, this work demonstrates conclusively that the surface syntax of an editor is surprisingly important from a human engineering point of view.

3. INPUT AND OUTPUT DEVICES AND TECHNIQUES

The work station used in text editing, generally called a terminal, consists of a keyboard used as an input device and a screen display or printer used as an output device. Screen display terminals may also be equipped with auxiliary input devices, such as a light pen or joystick, that serve to locate a specific item of information on the display. This section is a review of the psychological and human factors aspects underlying the design and use of keyboards, screen displays, and pointing devices.

A thorough discussion of the fundamentals of physiological psychology applicable to the design of computer terminals can be

found in RUPP77. Rupp and Hirsch review the mechanism of the human visual system and explain its optical properties. They are careful to consider the requirements of the entire population of terminal users, including persons with various visual disorders. They take into consideration ambient lighting and derive acceptable ranges for display color, refresh rate, brightness, contrast, and character quality. A similar treatment, starting with basic tactile parameters, is applied to keyboard design. Finally, the design of the entire work station, including chair, keyboard, and display height, is reviewed, and recommendations are provided for work-rest cycles and ambient conditions.

3.1 Key Entry

Key entry is unquestionably the most common means of encoding letters and numbers in computer-readable form. In addition to their use in interactive display terminals, keyboards are used in keypunches, key-to-disk and key-to-tape data-entry systems, photocomposers, hex keypads, pushbutton telephones, mail-sorting devices, and special-purpose operator consoles. In each case finger action is used to convert alphanumeric information to electronic form.

While there has been little research directly on the use of key entry in program and text editing, some of the information accumulated in over one hundred years of research on keyboard operations is directly applicable, and other results are suggestive. The earliest work was motivated by an interest in telegraphy, followed by decades of investigation of typing and adding machine operations. More recently, there have been a number of pertinent experiments on keying speed and error rate in electronic data entry, postal mail sorters, and telephone "dialing." The findings most applicable to program editing concern the design of keyboards, keying rates, and keying errors.

3.1.1 Keyboard Devices

The first recorded patent for a typewriter was taken out in England in 1714. Commercial models, based on a design of Sholes, Glidden, and Soule and manufactured by Philo Remington, did not become popular until the 1880s. The teletypewriter, a direct antecedent of today's interactive terminals, was invented in 1904, and the electric typewriter, as we know it today, came into use in about 1935.

Virtually all of the more than 10 million typewriters in use in the United States have the standard "QWERTY" arrangement of alphabetic keys, which is also duplicated on computer terminals and key entry devices. Many other arrangements of keys, designed for increased speed, have been proposed, but none have caught on [ALDE72]. On the

TABLE 7. SUMMARY OF PERFORMANCE[a]

	English Editor	*Notational Editor*
Mean percentage of editing tasks completed[b]		
Inexperienced users	42	28
Familiar users	63	43
Experienced users	84	74
Average	63	48
Mean percentage of erroneous commands[c]		
Inexperienced users	11	19
Familiar users	6.9	18
Experienced users	5.6	9.9
Average	7.8	16
Mean efficiency (percent)[d]		
Inexperienced users	43	31
Familiar users	53	36
Experienced users	58	53
Average	51	40

[a] Adapted from LEDG80.
[b] COMPLETION_RATE = (COR − ERR)/TOT_COR
[c] ERROR_RATE = (SYN + SEM)/NUM_CMDS
[d] EDITING_EFFICIENCY = (POS − NEG)/NUM_CMDS

where

COR	is the number of indicated corrections made to the text;
ERR	is the number of erroneous changes made to the text:
TOT-COR	is the total number of indicated corrections requested;
SYN	is the number of commands that were syntactically ill formed;
SEM	is the number of commands that were semantically meaningless;
NUM-CMDS	is the total number of commands issued;
POS	is the number of commands that resulted in an improvement of the text;
NEG	is the number of commands that resulted in a degradation of the text.

other hand, anarchy reigns with regard to punctuation keys and special-purpose keys (such as "backspace"). Various proposed standards are reviewed in SEIB72. One of the major problems is the putative desirability of tying the standard to the ASCII code in such a manner that the upper- and lowercase symbols represented by a given key differ only by a single bit in their ASCII representation (bit pairing).

In numeric keypads the controversy between proponents of the ten-key adding machine keyboard (1, 2, 3 in the bottom row) and the pushbutton telephone keyboard (1, 2, 3 in the top row, zero at the bottom) appears to favor the latter [SEIB72], although many new calculators still use the adding machine arrangement. Many computer terminals preserve the conventional two-hand numeric arrangement along the top row and also provide a separate cluster of numeric keys at the right of the keyboard.

The standard key size is 0.5 inch (1.27 centimeters) in diameter and the standard horizontal spacing is 0.75 inch (1.81 cm). The slope of the keyboard, the force necessary for key depression, the key displacement, and the type of kinesthetic feedback from key actuation may be varied within wide limits without affecting performance [ALDE72].

The familiar concept of the shift key may be extended to "chord" keyboards, of which the ultimate expression is the stenotype machine [HILL59]. The motivation behind multipress keyboards is that the number of keystrokes per second is limited by the ballistic constraints on the fingers rather than the information-processing limitations of the operator; hence mental encoding of the material into group codes may lead to greater throughput [ROCH78]. Engelbart and English have experimented with a five-key chord set mounted on a "mouse" used for cursor control. They claim that the chord set is favored by operators over the standard two-handed keyboard for entering literal strings of fewer than ten characters [ENGD68]. Seibel suggests that improvements of up to 150 percent in text entry rate are obtainable, albeit at a considerable training cost [SEIB72].

Functional encoding, where each individual key corresponds to a predefined string of characters, is another possible means of increasing throughput. Schoonard and Boies, among others, describe experiments with a "short-type" abbreviation scheme for the most common words in a particular type of text and report significant gains [SCHO75]. Because of the length of training required, these methods are unlikely to see widespread application in text editing. In any case, most commands in text editors, and most keywords and variable names in programming languages, are already quite short. Some terminals offer alternative programmable or switchable keyboard and display symbol sets, such as APL characters, while others comprise programmable function keys for which visual identification is usually provided in the form of disposable overlays. A more flexible means of encoding keys, where the operator must type in only the minimum number of letters necessary to provide an unambiguous operand and the computer then completes the rest of the message to provide an easily verifiable display, has also been studied [FIEL78] and is discussed in the following.

The human engineering aspects of the data communications protocols of typewriter-like terminals used for editing are considered by Ossanna and Saltzer [OSSA70]. They compare duplex and half-duplex connections in terms of desirable and undesirable sequencing of input and output and of asynchronous read-ahead and type-ahead strategies. They also discuss—without, however, presenting any experimental observations—the effects of locking and unlocking the keyboard, local and echo printing, line overflow conditions, interrupt signals (break key), and accidental disconnects. Listed among important terminal characteristics are device self-identifying features (still available only as an expensive option on most terminals) and programmed control of secondary features such as line feed, horizontal and vertical tabulators, character size and code, bit rate, parity, and audible whistles or bells. Much of the information presented in the paper is the direct result of the authors' experience with M.I.T.'s MULTICS and IBM's TSS.

Ossanna and Saltzer argue convincingly that users should not be unnecessarily prohibited from entering items as fast as they can think of and type them. Four mechanisms they suggest for accomplishing this are

(1) read-ahead or type-ahead operation of the terminal, whereby typing action characters triggers program execution but does not inhibit further input;
(2) allowance for more than one independent component between action characters—for example, multiple commands on a single line;
(3) no unnecessary activity by the command processor—arguments should be accepted immediately after a command rather than being necessarily prompted;
(4) provisions for the creation of data files or macros as an alternative to direct input from the keyboard.

They conclude that truly convenient terminal operation can be achieved only through coordinated design of the terminal, the terminal control hardware, the terminal control software, the system's command interpreter, the commands, and other programs.

Perhaps the most detailed consideration of the human factors aspects of typewriter-like computer terminals designed for time-sharing systems is that undertaken by Dolotta [DOLO70]. The specifications offered were initially generated by the Character-Oriented Conversational Terminal Subcommittee of the Time Sharing Project of SHARE and had been reviewed prior to publication by over one hundred "experts." Consequently, Dolotta may perhaps be pardoned if some of his remarks sound dogmatic and if he regards terminal development as an incessant struggle between innocent terminal users and callous terminal manufacturers.

Dolotta provides a list of required and optional features, including size, weight, power consumption, audible signals, documentation, maintainability, keyboard layout, forward and backward paper feed, inking mechanisms, noise and vibration characteristics, character spacing, size and shape, control panel layout, function keys, and communications capabilities. Figure 10 gives some idea of the functions considered. Interestingly, however, Dolotta did not foresee the advent of printing terminals with one-line or several-line displays, such as are currently available in several word-processing stations. Furthermore, some of the suggested functions, such as the mechanism for locking the keyboard, appear unnecessary in modern read-ahead terminals.

3.1.2 Typing Speed

Among the important physiological and psychological correlates of typing skill are finger ballistics, reaction time, motor learning, short-term memory, and human information-processing capability. The average single-finger tapping rate is of the order of six taps per second, with a 20 percent increase in speed from little finger to index finger and a 2–3 percent increase in favor of the dominant hand. Good typists average less than 0.2 second per keystroke (50 words per minute) for short periods, in contrast to 0.7 second per keystroke for less frequent keyboard users [ALDE72]. The interval between the fastest digrams is about 0.08 second for experienced typists, corresponding to a tapping rate of 12 taps per second [FOX64]. Digrams typed with alternate hands are about 25 percent faster than digrams typed with the same hand. These digram intervals determine the necessary output rate for a display under direct keyboard control. It is also of interest that locking the keyboard for a period almost as long as the average digram interval does not unduly interfere with typing performance after a short adaptation period [ALDE72].

Baddeley reports that it takes the average postal employee with no previous typing skill 60 hours of practice to reach a rate of 0.67 second per keystroke on material containing British postal codes (which may be more similar to program code than to plain text). His experiments show that the most "efficient" training regimen demands only a single hour of practice per day. If the schedule is accelerated, the total number of hours of practice required to reach criterion-level speed increases [BADD78]. In view of the ample evidence that poor typing

1. MASTER POWER ON/OFF SWITCH
2. STANDBY/ACTIVE SWITCH
3. LOCAL/REMOTE SWITCH
4. ATTENTION
5. INTERRUPT
6. NEW LINE
7. LINE FEED
8. REVERSE LINE FEED
9. HALF-LINE FEED
10. REVERSE HALF-LINE FEED
11. CARRIER RETURN
12. TAB
13. REVERSE TAB
14. VERTICAL TAB
15. REVERSE VERTICAL TAB
16. NEW PAGE
17. TAB SET
18. TAB CLEAR
19. TAB STOPS INDICATORS
20. VERTICAL TAB SET
21. VERTICAL TAB CLEAR
22. VERTICAL TAB STOPS INDIC.
23. LEFT MARGIN POSITION INDIC.
24. LEFT MARGIN SET
25. RIGHT MARGIN POSITION INDIC.
26. RIGHT MARGIN SET
27. MARGIN RELEASE
28. END LIGHT/RIGHT MARG. BELL
29. RIGHT MARGIN BELL INHIBIT
30. SHIFT
31. SHIFT LOCK
32. CARRIER POSITION INDICATOR
33. TOUCH CONTROL
34. COLOR SHIFT TO RED
35. COLOR SHIFT TO BLACK
36. COLOR SHIFT TO NO-INK
37. PAPER FEED ALARM RESET
38. PAPER FEED ALARM
39. PAPER FEED ALARM INHIBIT
40. AUTOMATIC REPEAT
41. AUTO. REPEAT PRESSURE ADJ.
42. MANUAL KEYBOARD UNLOCK
43. "PROCEED" INDICATOR LIGHT
44. "STOP" INDICATOR LIGHT
45. SINGLE/DOUBLE/TRIPLE SPACE
46. PRINT DENSITY
47. PLATEN PRESSURE ADJUSTMENT
48. PRINT ELEMENT SENSING
49. PRINT SUPPRESS ON/OFF SWITCH
50. COMM. LINE STATUS LIGHT
51. COMM. LINE "HANG-UP" SIGNAL
52. TERMINAL ID. ("HERE IS")
53. DATA CHECK INDICATOR
54. DATA CHECK INDICATOR RESET
55. DIAL-UP
56. TERMINAL SENSE DATA
57. EOM-AFTER-NEW-LINE
58. PROGRAMMABLE INDICATORS
59. NEAR-END-OF-PAGE ALARM
60. PAPER RELEASE
61. SPACE/BLANK
62. BACKSPACE
63. FULL/HALF-DUPLEX SWITCH
64. A, a, B, b, C, c, ... (i.e., all the printing ASCII codes not accounted for above)
65. ETB, EOT, ACK, ... (i.e., all the nonprinting ASCII codes not accounted for above)

FIGURE 10. Indicator functions for interactive terminals (adapted from DOLO70).

habits are difficult to shed and that most self-taught typists do not reach even half the speed expected from entry-level typists, it may be worth considering the benefits of specialized training for interactive computer users.

In a well-constructed and often cited experiment, Shaffer and Hardwick measured the short-term performance of twenty qualified touch typists at the University of Exeter. The material ranged from difficult but coherent text to randomly selected words and to random character sequences generated by zeroth- and first-order Markov processes. Very little difference was found between ordinary prose (average: 0.159 second per keystroke) and random words (0.162 second per keystroke), but the mean interval between symbols more than doubled for random letter strings broken into word-sized chunks [SHAF68]. This is entirely consistent with observations regarding the limitation of human information-processing capabilities [MILG56].

Explanations of the typist's ability to process meaningful segments more rapidly than random sequences usually involve the "acquisition of a hierarchy of habits" (in this case the ability to type a whole word as a single unit), which was first investigated in an inspiring turn-of-the-century study of novice, apprentice, and master telegraphers [BRYA99]. Another possible explanation is that typists may be able to read farther ahead (without forgetting the material) on words as opposed to random character strings [HERS65]. Other experiments on key-to-disk data entry with experienced keypunch operators, however, yielded a median interstroke interval of about 200 milliseconds regardless of the type of material—numerals, alphanumeric codes, or English words—as long as the "text" was broken up into groups of about five symbols [NEAL77].

Since the keystroke sequences in many editing tasks do not resemble ordinary prose, it would be of some interest to determine whether or not data derived from such transcription tasks are of any value in estimating parameters for keystroke models. A frequently verified observation that might transfer to experiments on text editing is that short-term timed tests generally yield estimates of productivity that are about twice those measured over an eight-hour workday [SEIB72]. Fatigue effects may account for this difference, since measurements indicate that experienced key entry operators may execute 56,000 to 83,000 keystrokes per day corresponding only to 0.51–0.35 second per keystroke [ALDE72].

3.1.3 Keying Errors

As in the case of typing rates, most of the published information on errors was not obtained in an editing environment. Furthermore, error rates vary much more from operator to operator and from task to task than does speed. Error rates of keypunch and bankproof machine operators at several installations were found to range from 0.02 to 0.04 percent [KLEM62]. Shaffer and Hardwick, in the typing experiment mentioned above, observed a 0.6 percent undetected error rate on prose and word material and 2.3 percent on random character strings. They characterized the errors as those of *omission*, *response*, *reading*, *context*, and *random*, and report the distribution among the various types [SHAF68]. Baddeley's postmen, at criterion speed, committed 1.0 percent errors on mail codes (possibly a strong argument for electronic mail) and immediately detected 50 percent of them. With a three-month layoff without practice the error rate doubled, while the speed decreased by only 30 percent [BADD78].

John Long studied the effects of visual feedback on keying performance. He showed that masking the keyboard reduces the speed and accuracy of skilled typists, while masking the printed text reduces only the accuracy. Masking the keyboard increased the error rate from 0.9 to 2.6 percent per character and decreased the speed by about 30 percent. Masking the copy increased the error rate by 40 percent because the operator failed to catch many errors [LONG76a].

Long also studied delayed irregular feedback by having the terminal print a symbol with a considerable delay (averaging 160 milliseconds) after the corresponding key was pressed. He showed that such delay affects only unskilled operators. Skilled operators tolerated the delayed auditory and

visual feedback with no degradation in performance after a brief training period [LONG76b]. Long's principal interest in these studies was the verification of psychological feedback models for highly practiced perceptual motor skills rather than improved device design.

In other typing experiments performed using a line-oriented text editor, the vast majority of the errors were noted and corrected before each line was entered; a much smaller number had to be corrected subsequently using the text editor. Furthermore, the residual word error rate after entry and correction of the text (a lengthy technical article) was 0.52 percent as compared with 3.40 percent in a five-minute timed test where no corrections were allowed [SCHO75]. The overall keystroke rate adopted by an individual of given skill is governed by the number of *detected* errors: if too many errors are incurred, the individual slows down [RABB70]. The specific trade-off point may be shifted by the reward structure imposed, which may stress either speed or error-free performance [ALDE72]. Hence we might expect that in text editing, commands—where a mistake might be disastrous—would be entered more hesitantly than text. The keying rate would also be governed by how conveniently errors may be corrected. The number of errors generally increases with word or code length; three or four characters appear to be the optimum group length [SEIB72].

Among different typists speed and error rate seem to be inversely correlated: the faster typists are more accurate [CARL63]. Among typists working approximately at the same speed, large differences in error rate are common: the most accurate 10 percent of the typists make six to ten times fewer mistakes than the least accurate 10 percent [SEIB72].

Different means of signaling detectable typing errors are reported in SEGA75. Seventy subjects performed two tasks (one consisting of entering 20 five-letter permutations and the other of listing 25 states) using a PLATO terminal. Erroneous entries were flagged either immediately or at the end of the task. It would appear that immediate interruption when an error is committed leads to about 25 percent faster task completion than the indication of errors only after completion of a major segment of the task. Other considerations, such as the total number of erroneous key presses, favor the delayed indication, but this result was not significant at the 0.05 level. The tasks and methods reported in this study are too specialized, however, to allow drawing any general conclusions regarding the most suitable form of error indication.

The relatively high rate of simple, "easily" detectable errors in keyed input gives rise to the question of automatic correction of such errors. A number of researchers, among whom C. R. Blair [BLAI60] is usually accorded precedence, have tackled the problem, and several commercially available word-processing systems actually incorporate spelling verification routines [PETE80]. Automatic error-correction techniques are described in MUTH77, which also contains many references to earlier experiments.

It is, of course, tempting to think of spelling correction as an integral part of editing. One cannot help asking to what level and complexity automatic error-correction techniques can be extended eventually. Currently available editors already provide some help of this type. Illegal commands and arguments are flagged immediately, although no automatic correction is provided. One level deeper, editors imbedded in conversational programming languages call on the interpreter to analyze a line of program code as soon as it is entered and provide a warning of incorrect syntax [WILC76]. Indications of execution errors may also be provided. The thought of immediate automatic correction of some of these errors, particularly in an interactive environment where the programmer can always override a miscorrection, is appealing [FOSD78].

3.2 Display Devices and Screen-Oriented Operations

Typewriter-oriented editors provide a keyboard as the user's sole means of communications with the system, but screen-oriented editors may provide additional dimensions for communication by means of graphic input devices. In this section we review the literature on display devices and

experimental observations on pointing devices and pointing skills, while adding our usual caveat that much of the information has been collected without regard for the specific requirements of editing tasks.

3.2.1 Display Format and Terminal Design

A comprehensive set of human factors guidelines for man/display interfaces is presented in ENGE75. According to Engel and Granda their major thrust was "to state guidelines based on observable, reported evidence gathered in some systematic manner rather than to rely on hearsay, personal preference, or programming convenience." Over 100 specific suggestions are made. Unfortunately, they are not related by citation to the bibliography of the report, and it is impossible to determine to what extent they meet the authors' objectives.

The suggestions are divided into the categories of *display format, frame content, command language, recovery procedures, user entry techniques, general principles,* and *response time requirements.* Some examples from the section on display formats that are applicable to editing are

- Display string of five or more digits or alphanumeric characters in groups of three or four.
- Number menu items starting with one, not zero. In counting, people start with one; in measuring, they start with zero.
- Use vertically assigned lists with left justification for most rapid scanning. Subclassifications can be identified by indenting.
- Use left justification with text, right justification for numerals.
- Always place a period after item selection number, at the end of a sentence, and where necessary for clarification.
- Make sure that abbreviations, mnemonics, and acronyms do not include punctuation.
- Use numbers only when listing selectable items. Alphabetic characters or bullets may be used in prose/text.

Pankove, in the introduction to a recent anthology on display design, lists the following principal psychophysical factors related to display perception [PANK80]: luminance and brightness; color (hue, saturation, brightness); contrast; directional visibility; and size and resolution. The minimum detectable visual stimulus consists of 60 quanta of blue-green (510 nanometers) light. Luminance is a measure of the radiation emitted by an object, while brightness takes into account the variation in the sensitivity of the human eye with wavelength. Since the human eye is capable of adapting to a $10^7 : 1$ range of light levels (with most of the adaptation taking place in the retina and only about $20 : 1$ in the pupil), the brightness requirements for a display depend primarily on the ambient illumination.

The smallest picture element need not be smaller than the size needed to subtend 1.3 minutes of arc, the effective resolution of the retina. This corresponds to 25 elements per centimeter at a viewing distance of 70 centimeters. Display contrast (the normalized difference between the brightest and least bright spots) ranges from about 3 to 20, and is adjustable for operator comfort on most display terminals. A gradual twofold change in contrast is normally imperceptible on large displays.

An ergonomic approach to the design of a specific display terminal is described in OLSO80. Olson claims that the optimal display character is 2.54 by 3.18 millimeters in size ($4 : 5$ aspect ratio) and is defined on a 7×11 dot matrix. His characters are surrounded by a one-dot-wide margin and lowercase characters have two-dot descenders for increased legibility. The display color is yellow-green on a grey background and a contrast level of $3 : 1$ is achievable even in very high ambient illumination. The cursor is a graphic rectangle alternating with the character occupying the same position at a rate sufficient to provide comfortable reading of the indicated character. The display-face tilt adjustment is -10 to $+30$ degrees relative to the vertical, and 90 degrees in horizontal rotation. The refresh rate is 60 hertz driven by a crystal clock to allow flicker-free operation on 50-hertz line frequency. A P-31 phosphor was selected as the best possible compromise to satisfy the ergonomic requirements of maximum retinal efficiency, maximum character sharpness, good display contrast, and minimum flicker without character smear.

The complex perceptual relationships between phosphor persistence, regeneration rate, display resolution, and scan order are examined in GOUL68 and DILL70. The conclusion of Gould and Dill is that the major effect of a pseudorandom scan order, as opposed to a raster scan, is to reduce the disturbing effects of display flicker when it does occur rather than to reduce substantially the regeneration rate required for flickerless display. They argue, on the basis of their experiments, that the minimum acceptable (flickerless) regeneration rate for computer-controlled cathode-ray-tube terminals with static displays is 15–20 frames per second.

Although most editor displays (except for one-line displays incorporated into typewriter terminals in word-processing units) are based on cathodoluminescence (cathode-ray terminals), Pankove lists light-emitting diodes (LED), plasma panels, electroluminescence (EL), incandescence, liquid crystals (LC), electrochromicity (ECD), electrophoresis (EPID), and electrooptic modulation as possible display mechanisms [PANK80].

For additional references on the ergonomic aspects of video terminal design, the reader may turn to CAKI79, which has a bibliography of 363 titles.

3.2.2 Pointing and Cursor Control Devices

Positional reference to a particular item displayed on a screen is one of the most common requirements in program editing and may be accomplished either by pointing directly at the item or by moving a cursor to the location. A cursor may be controlled either by the keyboard or by means of some ancillary analog positioning device. In normal editing operations, positional references are interspersed with alphanumeric entry of commands or text using the keyboard. The major design question, therefore, is to determine the most rapid, accurate, and convenient means of positional referencing.

A fairly complete description of the various devices available in 1975 for positional referencing is presented in RITC75. Ritchie considers the light pen, tracker ball, mouse, knee control, joystick, touch panel, and graphic tablet or coordinate digitizer. The article is most thorough in exploring the innumerable physical phenomena that may be exploited for curve tracing. No major new devices have become popular in the intervening six years, although several terminals are now equipped with X–Y wheels and technical improvements have taken place in light-pen and touch-panel design.

Engel and Granda also discuss the advantages and disadvantages of various cursor control devices without, however, specific reference to theoretical or experimental work supporting their views [ENGE75]. They consider the following devices:

(1) light pen (a light-sensitive device),
(2) selector pen (a light-emitting device),
(3) joystick,
(4) track ball,
(5) mouse,
(6) thumb wheels,
(7) digitizer stylus,
(8) keyboard.

Their overall recommendations favor the joystick on the basis of control characteristics and range of applications. They suggest that both rapid movement and vernier modes be made available under user control, and that the selectable areas on the screen be made as large as possible.

3.2.3 Pointing Skills

In a 1965 experiment English and his colleagues compared the speed and accuracy of positional referencing using a mouse, joystick, knee control, and light pen [ENGL67]. Since English et al. were specifically interested in editing, this experimental design took into consideration the "access time" of the devices, that is, the time necessary for the hand to leave the keyboard, execute the pointing task, and return to the keyboard. Both experienced and inexperienced subjects and both coarse and fine ("word" and "character") pointing tasks were studied. With the mouse, knee control, and joystick, the cursor was moved on the screen using visual feedback; with the light pen, the pointing action was direct.

The major conclusion of the experiment was that in the circumstances studied the mouse is the preferred device. The authors,

TABLE 8. COMPARISON OF POINTING DEVICES[a]

Experimental Condition	*Mean Time* (Seconds) *to Locate Operand*					
	Mouse	*Grafacon*	*Light Pen*	*Joystick* (absolute)	*Joystick* (rate)	*Knee Control*
Experienced subjects, "character mode," no penalty for errors	1.93	2.43	2.13	2.87	—	—
Experienced subjects, "character mode," 30 percent penalty for errors[b]	1.99	2.57	2.28	3.14	—	—
Experienced subjects, "word mode," no penalty for errors	1.68	1.92	1.81	1.99	—	—
Experienced subjects, "word mode," 30 percent penalty for errors	1.74	1.97	1.93	2.07	—	—
Inexperienced subjects, "character mode," no penalty for errors	2.62	3.26	2.43	3.29	5.22	2.36
Inexperienced subjects, "character mode," 30 percent penalty for errors	2.71	3.51	2.64	3.54	5.71	2.52

[a] Adapted from ENGL67, ©IEEE 1967.
[b] ENGL67 does not offer a rationale for the 30 percent error penalty.

however, were very careful to point out the restricted scope of their assumptions. The overall results are shown in Table 8; it is seen that the differences between the devices are minor. Unfortunately, the use of the keyboard for controlling the cursor was not included in this experiment.

In a subsequent experiment Goodwin compared the light pen, light gun, and keyboard in three cursor control tasks [GOOD75]. (The light gun is simply a pistol-grip mount for the light pen, with the somewhat awkward switch on the original pen replaced by a trigger. As it turned out, there was no significant observed performance difference between the light pen and the light gun, although the latter was better liked.) The three tasks consisted of pointing to randomly appearing spots on the screen, pointing to a series of spots in sequential top-to-bottom, left-to-right order, and pointing to typographic errors in a segment of text. Whenever the cursor reached the required position, the subject had to enter an "x" on the typewriter. Unfortunately, an x—even an uppercase X—can be entered with one hand; consequently the experiment did not really simulate the interspersed pointing and text entry typical of editing. Not surprisingly, the light pen/light gun proved faster than positioning by means of the very awkward keyboard arrangement provided (the spatial arrangement of the cursor control keys did not correspond to the direction of cursor movement, and some cursor motions required pressing the shift key as well). Nevertheless, the author reports that in the course of routine operations, persons who have either alternative available frequently use the keyboard. They tend to use the light pen mainly for reverse-direction operations, which in that particular keyboard require a multiple shift-key action. As also pointed out by English, the time required to reach a target with the light pen is independent of the initial position of the cursor and depends mainly on the final accuracy required (this assumes, of course, that the initial position of the light pen is independent of that of the target). On the other hand, with keyboard control the time is almost directly proportional to the length traveled by the cursor. In either case the size of the target is important: it is faster to point to a word than to a period.

It should also be mentioned that in both experiments the authors reported some dissatisfaction with the light pen with regard to accuracy, since signals are sometimes picked up from adjacent characters. Prolonged use of the light pen was also reported to be fatiguing since the arm cannot rest. The time necessary to pick up and deposit the light pen would be saved by using the touch panel, but the ordinary human index finger is not shaped correctly for selecting 1/8-inch characters. In one design mentioned by Ritchie the cursor position is offset from the sensed position of the finger

TABLE 9. COMPARISON OF CURSOR CONTROL DEVICES[a]

Device	Overall Times: Movement Time for Nonerror Trials (seconds): Homing Time M	Homing Time SD	Positioning Time M	Positioning Time SD	Total Time M	Total Time SD	Error Rate (%) M	Error Rate (%) SD	Summary of Models for Positioning Time (T_{pos})
Mouse	0.36	0.13	1.29	0.42	1.66	0.48	5	22	$T_{pos} = 1.033 + 0.096 \log_2(D/S + 0.5)$
Joystick	0.26	0.11	1.57	0.54	1.83	0.57	11	31	$T_{pos} = 1.036 + 0.205 \log_2(D/S + 0.5)$
Step keys	0.21	0.30	2.31	1.52	2.51	1.64	13	33	$T_{pos} = 1.197 + 0.052(D_x/S_x + D_y/S_y)$
Text keys	0.32	0.61	1.95	1.30	2.26	1.70	9	28	$T_{pos} = 0.658 + 0.209\, N_{min}$

[a] Adapted from CARD78b.

in such a manner that the finger can guide the cursor freely without obscuring it [RITC75].

The relative speeds of indirect pointing methods and keyboard-controlled cursors in selecting targets for editing tasks were eventually also compared in CARD78b. The pointing devices included in this experiment were the mouse and a rate-controlled isometric joystick (a peculiar choice since a direct-reading joystick was shown earlier, by a group which also included English, to be superior to the rate-controlled joystick). The key controls consisted of step keys (where the horizontal and vertical motions are independent of the information displayed) and text keys (which can cause the cursor to skip entire words or paragraphs). The light pen was not included in this experiment.

Learning effects which might favor one device or another were carefully eliminated by training all subjects to criterion. It was demonstrated, however, that the learning curves of positioning time versus amount of practice can be approximated by a power curve, as predicted in DEJO57.

Both positioning speed and error rate were studied in a comprehensive experimental design that allowed the study of these variables as a function of approach angle to the target, target distance from the initial position, and target size. Card and his colleagues demonstrate that the relation between positioning time, target size, and target distance obeys a version of Fitts' law [WELF68]. According to Fitts' law the time necessary to make a hand movement to a predetermined position may be expressed as

Positioning time

$$= K_0 + K \cdot \log_2(D/S + 0.5) \quad \text{seconds},$$

where D is the distance moved, S is the size of the target, and K and K_0 are constants.

Although Card computes the values of the constants K and K_0 under various conditions, for our purpose Table 9 provides a sufficient idea of the relative magnitudes of the phenomena involved and of the appropriate values of the parameters in Fitts' law. In Table 9, which shows values averaged over several experiments, homing time is measured from the time the subject's right hand leaves the space bar until the cursor begins to move. Positioning time is the interval between the beginning of cursor movement until the selection button is pressed. N_{min} is the minimum number of keystrokes necessary to reach the target with the text keys. The error rate is the fraction of unsuccessful trials in attempting to reach a target with an average size of 4.2 centimeters.

Card et al. conclude that the mouse is the uniformly superior device with respect to both speed and error rate. They also claim that the mouse approaches the physiological limits of performance for an analog pointing device. To credit these conclusions fully, however, more information than is contained in the paper would be required regarding the manner of computing the error rate, the exact configuration of the screen display, and the choice of cursor velocity under repeat-key action.

3.2.4 *Comparison of Key Entry and Menu Selection*

In a recent series of experiments with 32 subjects, Fields and her colleagues studied four methods of tactical data input simulating battlefield information requirements [FIEL78]. The four methods, all of which appear directly applicable to program and text editing, were (1) typing English words or codes; (2) typing with an autocorrection feature that attempted to correct transposition and single-character deletion, insertion, and substitution errors; (3) typing with an autocompletion feature that automatically completed nonambiguous entries and submitted them for verification to the operator; and (4) menu selection using a track ball (a stationary ball sunk in the console which can be rotated in any direction with the palm of the hand).

The lowest error rate was obtained through menu selection (Table 10). Only minor differences were obtained in entry rates, perhaps because the subjects were improving rapidly throughout the experiment. Autocompletion provided the fastest, as well as the least liked and most error-prone, means of data entry. The authors recommend the incorporation of menu selection schemes in tactical message communications systems, but urge replacement of the track ball with a light pen, touch panel, or typed code. They also recommend retention of the typing option for applications involving experienced operators and very long menus. They are not convinced of the value of automatic spelling correction in this application and believe that the autocompletion method is too confusing for novice operators. The experiment also provides incidental support for the importance of split screens for editing operations (although in this instance two separate display terminals were used for each operator).

TABLE 10. COMPARISON OF DATA ENTRY TECHNIQUES

Method	*Mean Number of Errors per Message*	*Mean Time (seconds) per Message*
Menus	2.64	397
Typing with error corrections	3.36	397
Typing	3.77	396
Typing with autocompletion	4.39	413

4. CONCLUSIONS

We have reviewed human-factors-oriented studies applicable to interactive text editors. In this section we first review behavioral aspects of text editors that have been explored only superficially or not at all and state our sense of important outstanding design issues. We then summarize progress in the areas that have been investigated with some degree of thoroughness.

4.1 Outstanding Design Issues

The amount of text, program code, and data accessible through interactive systems (this is the age of the information utility, the integrated office, the on-line management information system) is increasing rapidly, as is the number of people requiring daily or sporadic access to such information. Nevertheless we found no studies addressing the human factors aspects of the use of editors for searching, inspecting, or maintaining massive file systems; most of the activity centers instead on symbol manipulation in tasks of very limited scope where really significant improvement over the best of the present-day editors appears unlikely. We believe that as the boundaries between interactive editors, database query languages, and nonprocedural programming systems continue to fade, rich returns can be expected from research on the most appropriate conceptual organization of the vast amounts of information whose value is limited only by the user's patience and ability to extract the needed portions.

With regard to editors, the question can be posed in terms of the underlying model of information structure. For example, what are the behavioral advantages and disadvantages of editors incorporating nested subsections and structural information about the semantic content of files ("hypertext") as compared with conventional "flat-text" editors? A major unresolved issue is what constitutes the most convenient means of allowing the user combined access to parts of several files, or even of the same

file. Because file structures tend to vary among editors far more than commands applicable only within a file, even expert users may experience difficulties with file manipulation.

A related, but perhaps less important, problem is the evaluation of techniques for selecting small segments of text from a limited corpus (i.e., file). Competing methods include unique addresses (e.g., line numbers), content addressing (search pattern matching), and positional selection (i.e., pointing with cursor control or light pen). Questions to be answered include those of relative speed, error probability, and learnability of each technique.

The most suitable form of editor commands remains a subject of contention. Most editors, like most programming languages, tend to use prefix operators (print m1, m2), but some, including the popular UNIX editor, use postfix (m1, m2 print) or infix (m1, m2 move m3) notation. Command abbreviations and appropriate default options for operands also need more systematic investigation.

While most users familiar with both types of editors express a preference for two-dimensional screen-oriented editors over one-dimensional hard-copy-oriented editors, we have not seen any experiments comparing them with respect to productivity—although both varieties are available on the same CRT terminals under several systems, including CMS and UNIX.

The matter of appropriate notation for the specification and analysis of editors, while touched upon here and there, has not been satisfactorily dealt with and needs further investigation. Perhaps some of the current work on the formal design and verification of machine-to-machine communications protocols will also prove useful in the design of human–computer communications.

Only modest steps have been taken to study the nature of error feedback and the benefits of automatic error detection and error correction features.

Additional experimentation is needed on split-screen and multiple-screen editing operations. An important variable that does not appear to have been investigated is the screen size and the amount of material exposed to the user (who may be disoriented by frequent changes of screen). In current word-processing systems the amount of material displayed ranges all the way from half a line on a liquid crystal display to two entire pages of 60 or 70 lines. The most common display remains, however, the standard 24 by 80 CRT.

A very recent idea is the application of an optical scanner to manuscript editing [SUEN80]. A facsimile device is used to enter the rough draft and an accompanying transparent overlay of proofreading marks, and to print the edited output. The system can process cursive handwriting, hand printing, typed copy, line drawings, and continuous-tone pictures. When the input is not machine readable, the system performs the indicated additions, deletions, and rearrangements, producing a "clean" version. With machine-readable input the characters are encoded by an optical character recognition (OCR) subsystem in a form suitable for further editing using either the same system or a conventional editor. A graphics subsystem produces finished versions of hand-drawn sketches, including lettering; photographs may be scaled and inserted in the text. The concept of a scanner–computer combination necessitating only pencil and paper instead of a keyboard terminal clearly opens an entirely new dimension for human factors research on "interactive" editing.

Other technological advances that might enhance editing are color displays (and printers), audio input, and audio feedback. All of these are at the stage where serious consideration should be given to their application. Not at the same stage of readiness, but also within the realm of possibility, is the direct use of eye movement for pointing and menu selection.

Although text editors are generally considered simpler than procedure-oriented high-level programming languages, they are less standardized. Practically every installation boasts its own editor. It is our hope that as editor design gradually becomes more of an engineering-oriented discipline with a solid knowledge base buttressing design decisions, rational standardization

will lead toward improved portability of interactive editing skills.

4.2 Summary of Results

Here we review the findings that appear to us most likely to influence the design and evaluation of interactive text editors.

- Keystroke models can predict task time for expert users performing routine tasks with an accuracy comparable to individual variations between subjects. This accuracy is achieved, however, under the simplifying assumption that users perform editing tasks perfectly—without error—and thus represents an upper bound on how well an expert user might perform (Section 1.1).
- For an expert individual user the choice among alternative methods for performing routine editing tasks can be predicted with reasonable accuracy by means of a few simple selection rules as suggested by the GOMS model. Unless the sequence of commands used to accomplish an editing task can be predicted nearly perfectly, however, the level of detail at which editing is analyzed appears to be of little consequence (Section 1.2).
- Observations of subjects performing editing tasks in a controlled experimental environment reveal that error detection and correction, suboptimal choice of editing methods, and unpredictable mental activities account for between 25 and 50 percent of the task time. About as much difference between expert users can be attributed to user variability (mostly in error rate) as can be attributed to differences between editors (Section 1.3).
- Flexibility and options tend to increase the rate at which expert users can accomplish editing tasks but tend to reduce the rate for beginners (Section 2.2).
- Editor surface syntax, such as familiar, descriptive, English-like phrases versus the arcane notation found in some command languages, can increase editing efficiency and reduce task completion time and error rate. Data from one experiment indicate that these effects are more pronounced for users without previous experience with editors (Section 2.4).
- The various studies of editors ranging from timed tests to formal analysis of states and to the effects of surface syntax indicate, however, that there is less than a 2 : 1 difference between editors in common use today. Most users can perform about equally well on any reasonable editor (Sections 1 and 2).
- Ambient conditions for professional keyboard operators, including the effects of temperature, noise, work station layout, illumination, and work–rest cycles, have been extensively studied and the results probably apply to editing as well (Section 3).
- The ergonomic aspects of keyboard layout and design are well understood. Keying rates cannot be significantly improved without specialized and lengthy training, but there are established training rules for developing typing speeds of the order of 0.2 second per keystroke on short tasks. Productivity for an eight-hour day is about half that predicted from peak rates. Variations in keying speed and error rate are predictable as a function of input material as are speed/error trade-offs as a result of the reward structure. Individual variability among expert keyboard operators is far greater with regard to error rate than to speed. Although some consider type-ahead capability important for interactive editing, it increases the error rate for text and data entry (Section 3.1).
- Considerable data have also been accumulated on the design of display terminals, and appropriate values have been established for display contrast, color, character size, character shape, refresh rate, and screen orientation. Results on display-format design are less definitive, but it appears that menu selection is less error prone (but slower) than direct item entry. The emphasis on appropriate formatting of displayed values indicates that human factors consideration can be ignored only at the peril of considerable performance degradation (Section 3.2).
- Among display selection mechanisms the mouse appears to be nearly optimal in terms of pointing skill and accuracy, but its performance is only marginally superior to that of several other commonly accepted

cursor control mechanisms. For some tasks keyboard control of the cursor should be retained (Section 3.2).

As is happily always the case, much more remains to be done than has already been accomplished. Text editors represent the principal interface with computers for many people and therefore deserve a concerted effort toward applying psychological and human factors concepts and methods to increase their usability. Consumerism in the field of computers is here to stay.

ACKNOWLEDGMENTS

The authors gratefully acknowledge several stimulating discussions with Professor John Flowers of the Psychology Department of the University of Nebraska–Lincoln. The authors also wish to thank three anonymous reviewers and the editor of this special issue for a great many perspicacious suggestions and for several valuable literature references.

Table 1 is adapted from "The Keystroke-Level Model for User Performance Time with Interactive Systems," by S. K. Card, T. P. Moran, and A. Newell, in *Commun. ACM* **23,** 7 (July 1980). Table 2 is adapted from "A Procedure for Predicting Program Editor Performance from the User's Point of View," by D. W. Embley, M. T. Lam, D. W. Leinbaugh, and G. Nagy, in *Int. J. Man-Mach. Stud.* **10,** 6 (Nov. 1978), with permission from *International Journal of Man-Machine Studies,* ©Academic Press (London), Ltd. Figure 3 is adapted from "Studies in the Psychology of Computer Text Editing Systems," by S. K. Card, Ph.D. dissertation, Dep. Psychology, Carnegie-Mellon Univ., Pittsburgh, Pa., 1978; Xerox Research Rep. SSL-78-1, Palo Alto Research Center, Palo Alto, Calif., Aug. 1978. Table 3 is adapted from "Computer Text Editing: An Information-Processing Analysis of a Routine Cognitive Skill," by S. K. Card, T. P. Moran, and A. Newell, in *Cognitive Psychol.* **12** (1980); ©Academic Press. Tables 4 and 6 are adapted and Figure 9 is reproduced from "Evaluation of Computer Text Editors," by T. L. Roberts, Ph.D. dissertation, Dep. Computer Science, Stanford Univ., Stanford, Calif., 1979; Xerox Research Rep. SSL-79-9, Palo Alto Research Center, Palo Alto, Calif., 1979. Figure 6 is adapted from "User Engineering Principles for Interactive Systems," by W. J. Hansen, in *Proc. Fall Joint Computer Conf.,* vol. 39, AFIPS Press, Arlington, Va., 1971. Figure 7 is from "The Annotated Assistant; A Step Towards Human Engineering," by A. Singer, H. Ledgard, and J. Hueras, Tech. Rep. Dep. Computer and Information Science, Univ. Massachusetts, Amherst, 1977. Figure 8 is from "Formal Grammar and Factors Design of an Interactive Graphics System," by P. Reisner, in *IEEE Trans. Softw. Eng.* **SE-7,** 2 (March 1981), 229–240; ©IEEE 1981. Table 5 is adapted from "Formal Analysis of Program Editing," by P. Anandan, Master's thesis, Dep. Computer Science, Univ. Nebraska, Lincoln, Aug. 1979. Table 7 is adapted from "The Natural Language of Interactive Systems," *Commun. ACM* **23** 10 (Oct. 1980). Figure 10 is adapted from "Functional Specifications for Typewriter-like Time-Sharing Terminals," by T. A. Dolotta, in *Comput. Surv.* **2,** 1 (March 1970). Table 8 is adapted from "Display-Selection Techniques for Text Manipulation," by W. K. English, D. C. Engelbart, and M. L. Berman, in *IEEE Trans. Hum. Factors Electron.* **HFE-8,** 1 (March 1967), 5–15; ©IEEE 1967. Table 9 is adapted from "Evaluation of Mouse, Rate-Controlled Isometeric Joystick, Step Keys, and Text Keys for Text Selection on a CRT," by S. K. Card, W. K. English, and B. J. Burr, in *Ergonomics* **21** (1978), 601–613.

REFERENCES

ABRU56 ABRUZZI, A. *Work, workers, and work measurement,* Columbia University Press, New York, 1956.

ALDE72 ALDEN, D.G., DANIELS, R.W., AND KANARICK, A.F. "Keyboard design and operation: A review of the major issues," *Hum. Factors* **14,** 4 (1972), 275–293.

ANAN79 ANANDAN, P. "Formal analysis of program editing," Master's thesis, Dep. Computer Science, Univ. Nebraska, Lincoln, Aug. 1979.

ANAN80 ANANDAN, P., EMBLEY, D.W., AND NAGY, G. "An application of file-comparison algorithms to the study of program editors," *Int. J. Man-Mach. Stud.* **13** (1980), 201–211.

AUGM75 AUGMENTATION RESEARCH CENTER *NLS-8 command summary,* Stanford Research Inst., Menlo Park, Calif., May 1975.

BADD78 BADDELEY, A.D., AND LONGMAN, D.J.A. "The influence of length and frequency of training session on the rate of learning to type," *Ergonomics* **21,** 8 (1978), 627–635.

BING76 BINGHAM, H.W. "Text editing using APL/700," in *Proc. APL 76 Conference* (ACM), G.T. Hunter, Ed., 1976, pp. 78–82.

BLAI60 BLAIR, C.R. "A program for correcting spelling errors," *Inf. Control* **3,** 1 (March 1960), 60–67.

BOEH71 BOEHM, B.W., SEVEN, M.J., AND WATSON, R.A. "Interactive problem-solving—An experimental study of 'lockout' effects," in *Proc. Spring Joint Computer Conf.,* AFIPS Press, Arlington, Va., 1971, pp. 205–210.

BOLT73 BOLT BERANEK AND NEWMAN, INC. *TENEX test editor and corrector,* Manual DEC10-NG2FB-D, Cambridge, Mass., 1973.

BRYA99 BRYAN, W. L., AND HARTER, N. "Studies on the telegraphic language. The acquisition of a hierarchy of habits," *Psychol. Rev.* **6** (1899), 345–375.

CAKI79 CAKIR, A., HART, D.J., AND STEWART, T.F.M. *The VDT/manual,* Inca-Fiej Research Assoc., Darmstadt, W. Germany, 1979.

CARB68 CARBONELL, J.R., ELKIND, J.I., AND NICKERSON, R.S. "On the psychological importance of time in a time-sharing system," *Hum. Factors* **10,** 2 (1968), 135–142.

CARD76 CARD, S.K., MORAN, T.P., AND NEWELL, A. "The manuscript editing task: A routine cognitive skill," Xerox Research Rep. SSL-76-8, Palo Alto Research Center, Palo Alto, Calif., Dec. 1976.

CARD78a CARD, S.K. "Studies in the psychology of computer text editing systems," Ph.D. dissertation, Dep. Psychology, Carnegie-Mellon Univ., Pittsburgh, Pa., 1978; also Xerox Research Rep. SSL-78-1, Palo Alto Research Center, Palo Alto, Calif., Aug. 1978.

CARD78b CARD, S.K., ENGLISH, W.K., AND BURR, B.J. "Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT," *Ergonomics* **21** (1978), 601–613.

CARD80a CARD, S.K., MORAN, T.P., AND NEWELL, A. "Computer text editing: An information-processing analysis of a routine cognitive skill," *Cognitive Psychol.* **12** (1980), 32–74.

CARD80b CARD, S.K., MORAN, T.P., AND NEWELL, A. "The keystroke-level model for user performance time with interactive systems," *Commun. ACM* **23,** 7 (July 1980), 396–410.

CARL78 CARLS, C.B. "Getting ready for word processing's second generation," *Datamation* **24,** 9 (Sept. 1978), 139–144.

CARL63 CARLSON, G. "Predicting clerical error in an EDP environment," *Datamation* **9,** 2 (Feb. 1963), 34–36.

CHIN78 CHIN-CHANCE, S.A. "A mathematical model of word recognition strategies," Ph.D. dissertation Dep. Educational Psychology, Univ. of Hawaii, Honolulu, 1978.

COUL76 COULOURIS, G.F., ET AL. "The design and implementation of an interactive document editor," *Softw. Pract. Exper.* **6** (April–June 1976), 271–279.

DEJO57 DEJONG, J.R. "The effect of increasing skill on cycle time and its consequences for time standards," *Ergonomics* **1** (1957), 51–60.

DEUT67 DEUTSCH, P.L., AND LAMPSON, B.W. "An online editor," *Commun. ACM* **10,** 12 (Dec. 1967), 793–803.

DEVO67 DEVOE, D.B. "Alternatives to handprinting in the manual entry of data," *IEEE Trans. Hum. Factors Electron.* **HFE-8,** 1 (1967), 21–31.

DILL70 DILL, A.B., AND GOULD, J.D. "Flickerless regeneration rates for CRT displays as a function of scan order and phosphor persistence," *Hum. Factors* **12,** 15 (1970), 465–471.

DOLO70 DOLOTTA, T.A. "Functional specifications for typewriter-like time-sharing terminals," *Comput. Surv.* **2,** 1 (March 1970), 5–31.

DREY55 DREYFUS, H. *Designing for people,* Simon and Schuster, New York, 1955.

DZID78 DZIDA, W., HERDA, S., AND ITZFELDT, W.D. "User-perceived quality of interactive systems," *IEEE Trans. Softw. Eng.* **SE-4,** 4 (July 1978), 270–276.

EMBL78 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," *Int. J. Man–Mach. Stud.* **10,** 6 (Nov. 1978), 639–650.

EMBL81a EMBLEY, D.W., AND NAGY, G. "Empirical and formal methods for the study of computer editors," in *Computing skills and the user interface,* M. Coombs, Ed., Academic Press, London, 1981.

EMBL81b EMBLEY, D.W., AND NAGY, G. "SIMPLE, a friendly programming environment for beginners and casual users," Dep. of Computer Science Rep., Univ. Nebraska, Lincoln, 1981.

ENGE75 ENGEL, E., AND GRANDA, R.E., "Guidelines for man/display interfaces," IBM Poughkeepsie Lab. Tech. Rep. TR 00.02720, Poughkeepsie, N.Y., Dec. 1975.

ENGD68 ENGELBART, D.C., AND ENGLISH, W.K. "A research center for augmenting human intellect," in *Proc. 1968 Fall Joint Computer Conf.* AFIPS Press, Arlington, Va., pp. 395–410.

ENGL67 ENGLISH, W.K., ENGELBART, D.C., AND BERMAN, M.L. "Display-selection techniques for text manipulation," *IEEE Trans. Hum. Factors Electron.* **HFE-8,** 1 (March 1967), 5–15.

FIEL78 FIELDS, A.F., MALSANO, R.E., AND MARSHALL, C.F. "A comparative analysis of methods for tactical data inputting," Rept. AD-A060562/6, U.S. Army Research Institute for Behavioral and Social Sciences, Sept. 1978.

FOSD78 FOSDICK, L.D. "Detecting errors in programs," in *Proc. IFIP Working Conf. Performance Evaluation of Numerical Software,* Baden, Austria, Dec. 1978.

FOX64 FOX, J.G., AND STANSFIELD, R.G. "Digram keying times for typists," *Ergonomics* **7,** 3 (July 1964), 317–320.

FREE66 FREEDMAN, J.L., AND LANDAUER, T.K. "Retrieval of long-term memory: Tip-of-the-tongue phenomenon," *Psychol. Sci.* **4,** 8 (1966), 309–310.

GAIN78 GAINES, B. "Programming interactive dialog," in *Pragmatic programming and sensible software,* Online Conferences Ltd., Uxbridge, England, 1978.

GILB77 GILB, T., AND WEINBERG, G.M. *"Humanized input techniques for reliable keyed input,"* Winthrop, Cambridge, Mass., 1977.

GOOD75 GOODWIN, N.C. "Cursor positioning on

an electronic display using lightpen, lightgun, or keyboard for three basic tasks," *Hum. Factors* **17**, 3 (June 1975), 289–295.

GOOD78 GOODMAN, T., AND SPENCE, R. "The effect of systems response time on interactive computer-aided problem solving," in *Proc. ACM SIGGRAPH 78* (ACM), 1978, pp. 100–104.

GOUL68 GOULD, J.D. "Visual factors in the design of computer-controlled CRT displays," *Hum. Factors* **10**, 4 (Aug. 1968), 359–376.

GRIG70 GRIGNETTI, M.C., AND MILLER, D.C. "Modifying computer response characteristics to influence command choice," in *Proc. IEE Conf. on Man–Machine Interaction*, Teddington, England, Sept. 1970, pp. 201–205.

GROS76 GROSSBERG, M., WIESEN, R.A., AND YNTEMA, D.B. "An experiment on problem solving with delayed computer responses," *IEEE Trans. Syst., Man, Cybern.* **SMC-6** (March 1976), 219–222.

HALS77 HALSTEAD, M.H. *Elements of software science*, North Holland, New York, 1977.

HAMM79 HAMMER, J.M., AND ROUSE, W.B. "Analysis and modeling of freeform text editing behavior," in *Proc. 1979 Internat. Conf. Cybernetics and Society*, Denver, Col., Oct. 1979.

HANS71a HANSEN, W.J. "User engineering principles for interactive systems," in *Proc. Fall Joint Computer Conf.*, vol. 39, AFIPS Press, Arlington, Va., 1971, pp. 523–532.

HANS71b HANSEN, W.J. "Creation of hierarchic text with a computer display," Ph.D. dissertation, Dep. Computer Science, Stanford Univ., Stanford, Calif., June 1971.

HANS78 HANSEN, W.J., DORING, R., AND WHITLOCK, L.R. "Why an examination was slower on-line than on paper," *Int. J. Man–Mach. Stud.* **10** (1978), 507–519.

HAZE80 HAZEL, P. "Development of the ZED text editor," *Softw. Pract. Exper.* **10**, 1 (Jan. 1980), 57–76.

HECK78 HECKEL, P. "A technique for isolating differences between files," *Commun. ACM* **21**, 4 (Apr. 1978), 264–268.

HERS65 HERSHMAN, R.L., AND HILLIX, W.A. "Data processing in typing: Typing rate as a function of kind of material and amount exposed," *Hum. Factors* **7** (Oct. 1965), 483–492.

HILL59 HILL, H.H. *Machine touch shorthand theory*, Touch Shorthand Academy, Boston, Mass., 1959.

IBM76 IBM *CMS user's guide, release 3*, GC20-1818-0, March 1976.

IBM78 IBM "IBM Waterloo interactive data job entry terminal," Rep. SB30-1139-1, 1978.

INFO79 INFOTECH INTERNATIONAL *Man/computer communications*, Infotech State of the Art Report, vols. 1 and 2, Maidenhead, England, 1979.

IVER62 IVERSON, K.E. *A programming language*, Wiley, New York, 1962.

JONE78 JONES, P.F. "Four principles of man-computer dialog," *IEEE Trans. Prof. Commun.* **PC-21** (Dec. 1978), 154–159.

KEME71 KEMENY, J.G., AND KURTZ, T.E. *BASIC programming*, Wiley, New York, 1971.

KENN75 KENNEDY, T.C.S. "Some behavioral factors affecting the training of naive users of an interactive computer system," *Int. J. Man–Mach. Stud.* **7** (1975), 817–834.

KERN79 KERNIGHAN, B.W. "Advanced text editing on UNIX," Bell Labs. Tech. Rep., Murray Hill, N.J., Feb. 1979.

KLEM62 KLEMMER, E.T., AND LOCKHEAD, G.R. "Productivity and errors in two keying tasks: A field study," *J. Appl. Psychol.* **46** (1962), 401–408.

LACH79 LACHMAN, R., LACHMAN, J.L., AND BUTTERFIELD, E.C. *Cognitive psychology and information processing*, Erlbaum, Hillsdale, N.J., 1979.

LEDG78 LEDGARD, H.F., AND SINGER, A. "Formal definition and design," COINS Tech. Rep. 78-01, Univ. Massachusetts, Amherst, Feb. 1978.

LEDG80 LEDGARD, H., WHITESIDE, J.A., SINGER, A., AND SEYMOUR, W. "The natural language of interactive systems," *Commun. ACM* **23**, 10 (Oct. 1980), 556–563.

LEDG81 LEDGARD, H., SINGER, A., AND WHITESIDE, J.A. *Directions in human factors for interactive systems*, Springer-Verlag, New York, 1981.

LONG76a LONG, J. "Visual feedback and skilled keying: Differential effects of masking the printed copy and the keyboard," *Ergonomics* **19**, 1 (1976), 93–110.

LONG76b LONG, J. "Effects of delayed irregular feedback on unskilled and skilled keying performance," *Ergonomics* **19**, 2 (1976), 183–202.

MART73 MARTIN, J. *Design of man-computer dialogue*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

MILG56 MILLER, G.A. "The magical number seven, plus or minus two: Some limits on our capacity for information processing," *Psychol. Rev.* **63**, 2 (March 1956), 81–97.

MILL77a MILLER, L.A., AND THOMAS, J.C. JR. "Behavioral issues in the use of interactive systems," *Int. J. Man–Mach. Stud.* **9** (1977), 509–536.

MILL77b MILLER, L.H. "A study in man–machine interaction," in *Proc. 1977 Nat. Computer Conf.*, AFIPS Press, Arlington, Va., pp. 409–421.

MILR68 MILLER, R.B. "Response time in man-computer conversational transactions," in *Proc. 1968 Fall Joint Computer Conf.*, AFIPS Press, Arlington, Va., pp. 267–277.

MORA81 MORAN, T.P. "The command language grammar: A representation for the user interface of interactive computer sys-

tems," *Int. J. Man-Mach. Stud.* **14** (1981), in press.

MUTH77 MUTH, F.E., JR., AND THARP, A.L. "Correcting human error in alphanumeric terminal input," *Inf. Process. Manage.* **13,** 6 (1977), 329–337

NEAL77 NEAL, A.S. "Time intervals between keystroke records and fields in data entry with skilled operators," *Hum. Factors* **19,** 2 (Apr. 1977), 163–170.

NEWM77 NEWMAN J. "The processing of two types of command statement: A contribution to cognitive ergonomics," *IEEE Trans. Syst., Man, Cybern.* **SMC-7,** 12 (Dec. 1977), 871–875.

OLSO80 OLSON, N. *Ergonomics of the SPERRY UNIVAC UTS-4000 System visual display unit,* Sperry Univac Worldwide Industrial Design Center, Blue Bell, Pa., 1980.

OREN74 OREN, S.S. "A mathematical theory of man–machine text editing," *IEEE Trans. Syst., Man, Cybern.* **SMC-4,** 3 (May 1974), 258–267.

OSSA70 OSSANNA, J.F., AND SALTZER, J.H. "Technical and human engineering problems in connecting terminals to a time-sharing system," in *Proc. 1970 Fall Joint Computer Conf.*, vol. 37, AFIPS Press, Arlington, Va., pp. 355–362.

OSSA77 OSSANNA, J.F. *NROFF/TROFF user's manual,* Bell Labs. Tech. Rep., Murray Hill, N.J., May 1977.

PANK80 PANKOVE, J.I. *Display devices,* Springer-Verlag, Secaucus, N.J., 1980.

PANS77 PANSOPHIC SYSTEMS, INC. *Panvalet user reference manual OSUP10-7712,* Oakbrook, Ill., 1977.

PETE80 PETERSEN, J.L. "Computer program for detecting and correcting spelling errors," *Commun. ACM* **23,** 12 (Dec. 1980), 676–687.

RABB70 RABBITT, P.M.A., AND VYAS, S.M. "An elementary preliminary taxonomy for some errors in laboratory choice RT tasks," in *Attention and performance III,* A.F. Sanders, Ed., North-Holland, Amsterdam, 1970, pp. 56–76.

REIM78 REIMHEN, G.W. "Automated text-editing (a bibliography with abstracts)," NTIS/PS78/0391/9, Apr. 1978.

REIS81 REISNER, P. "Formal grammar and factors design of an interactive graphics system," *IEEE Trans. Softw. Eng.* **SE-7,** 2 (March 1981), 229–240.

RIDD76 RIDDLE, E.A. *Comparative study of various text editors and formatting systems,* Air Force Data Services Center, Washington, D.C., Aug. 1976 (AD-AO29 050).

RITC75 RITCHIE, G.J., AND TURNER, J.A. "Input devices for interactive graphics," *Int. J. Man-Mach. Stud.* **7,** 5 (Sept. 1975), 639–660.

ROBE79 ROBERTS, T.L. "Evaluation of computer text editors," Ph.D. dissertation, Dep. of Computer Science, Stanford Univ., Stanford, Calif., Nov. 1979; also Xerox Research Rep. SSL79-9, Palo Alto, Calif., 1979.

ROCH78 ROCHESTER, N., BEQUAERT, F.C., AND SHARP, E.M. "The chord keyboard," *Computer* **11,** 12 (Dec. 1978), 57–63.

ROUS75 ROUSE, W.B. "Design of man-computer interface for on-line interactive systems," *Proc. IEEE* **63,** 6 (June 1975), 847–857.

RUPP77 RUPP, B.A., AND HIRSCH, R.S. "Human factors of workstations with display terminals," Rep. HFC-22, Human Factors Center, IBM, General Products Div., San Jose, Calif., Nov. 1977.

SAND78 SANDEWALL, E. "Programming in the interactive environment: The 'LISP' experience," *Comput. Surv.* **10,** 1 (March 1978), 35–71.

SAVI69 SAVITSKY, S. "Son of STOPGAP," Rep. SAILON 50.1, Stanford Artificial Intelligence Lab., Stanford, Calif., 1969.

SCHO75 SCHOONARD, J.W., AND BOIES, S.J. "Short-type: A behavioral analysis of typing and text entry," *Hum. Factors* **17,** 2 (Apr. 1975), 203–214.

SEGA75 SEGAL, B.Z. "Effects of error interruption on student performance at interactive terminals," Tech. Rep. UIUCDS-R-75-727, Dep. Computer Science, Univ. Illinois, Urbana/Champaign, May 1975.

SEIB75 SEIBEL, R. "Data entry devices and procedures," in *Human engineering guide to equipment design,* H.P. VanCott and R.G. Kinkade, Eds., U.S. Government Printing Office, Washington, D.C., 1972, pp. 311–344.

SHAF68 SHAFFER, L.H., AND HARDWICK, J. "Typing performance as a function of text," *Q. J. Exp. Psychol.* **20,** 4 (1968), 360–369.

SHNE79 SHNEIDERMAN, B. "Human factors experiments in designing interactive systems," *Computer* **12,** 12 (Dec. 1979), 9–19.

SHNE80 SHNEIDERMAN, B. *Software psychology,* Winthrop, Cambridge, Mass., 1980.

SING77 SINGER, A., LEDGARD, H., AND HUERAS, J. "The annotated assistant: A step towards human engineering," Tech. Rep., Dep. Computer and Information Science, Univ. Massachusetts, Amherst, 1977.

STAN75 STANFORD CENTER FOR INFORMATION PROCESSING *Wylbur/370: The Stanford timesharing system reference manual,* 3rd ed., Stanford Univ., Stanford, Calif., Nov. 1975.

SUEN80 SUENAGA, Y., AND NAGURA, M. "A facsimile based manuscript layout and editing system by auxiliary mark recognition," in *Proc. 5th Internat. Conf. Pattern Recognition,* IEEE Computer Society, Dec. 1980, pp. 856–858.

TECO69 *Text editor and correction reference manual,* Interactive Sciences Corp., Braintree, Mass., 1969.

TEIT79 TEITELBAUM, T. "The Cornell program synthesizer: A syntax-directed programming environment," *SIGPLAN Notices* **14,** 10 (Oct. 1979).

TEIW79 TEITELMAN, W. "A display oriented programmer's assistant," *Int. J. Man-Mach. Stud.* **11,** 2 (March 1979), 157–187.

THOR34 THORNDIKE, E.L., AND ROCK, R.T., JR. "Learning without awareness of what is being learned or intent to learn it," *J. Exper. Psychol.* **17,** 1 (1934).

TREU75 TREU, S. "Interactive command language design based on required mental work," *Int. J. Man-Mach. Stud.* **7** (1975), 135–149.

UNCN79 UNIVERSITY OF NEBRASKA COMPUTER NETWORK *A guide to NUROS*, Lincoln, Neb., 1979.

VAND71 VANDAM, A., AND RICE, D.E. "On-line text editing: A survey," *Comput. Surv.* **3,** 3 (Sept. 1971), 93–114.

WALT74 WALTHER, G.H., AND O'NEIL, H.F., JR. "On-line user-computer interface—The effect of interface flexibility, terminal type, and experience on performance," in *Proc. 1974 Nat. Computer Conf.*, vol. 43, AFIPS Press, Arlington, Va., pp. 379–384.

WANG78 *Wang word processor operator's guide,* 3rd release, Wang Laboratories, Inc., Lowell, Mass., 1978.

WASS73 WASSERMAN, A.I. "The design of idiot-proof interactive systems," in *Proc. 1973 Nat. Computer Conf.,* vol. 42, AFIPS Press, Arlington, Va., pp. M34–M38.

WEIN71 WEINBERG, G.M. *The psychology of computer programming,* Van Nostrand Reinholt, New York, 1971.

WELF68 WELFORD, A.T. *Fundamentals of skill,* Methuen, London, 1968.

WELF76 WELFORD, A.T. *Skilled performance: Perceptual and motor skills,* Scott, Foresman, Glenview, Ill., 1976.

WILC76 WILCOX, T.R., DAVIS, A.M., AND TINDALL, M.H. "The design and implementation of a table driven, interactive diagnostic programming system," *Commun. ACM* **19,** 11 (Nov. 1976), 609–616.

YULE72 YULE, A. "Human response times in a graphic environment," *Comput. Bull.* **16** (June 1972), 304–305.

RECEIVED MARCH 1980; FINAL REVISION ACCEPTED JANUARY 1981