Palo Alto Research Center

Studies In The Psychology Of Computer Text Editing Systems

By Stuart K. Card

XEROX

STUDIES IN THE PSYCHOLOGY OF COMPUTER TEXT EDITING SYSTEMS

BY STUART K. CARD

AUGUST 1978

SSL-78-1

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

ABSTRACT

Six studies of user interaction with on-line computer text editing systems are reported. Editing times for benchmark tasks on several editing systems were collected to gauge the range of performance across systems. Using the measured times, it was possible to predict when an editing system would outperform a typewriter. To model the user's behavior in greater detail, an information processing model of editing performance is proposed describing the user's "goals", "operators", "methods", and "selection rules". An important issue in such a model is how the model's accuracy depends on the grain of analysis. To find out, the model was recast at nine different levels of grain size and the accuracy of the different versions compared. From observations of users on several different systems, it was discovered that on each task, the users go through a similar sequence of task assimilation, target location, target modification, and verification. This concept of a "unit task cycle" was used to predict rough performance times for a proposed system prior to system specification. With respect to the target location part of a task, four devices for pointing to a target were compared and modeled. Using Fitts's Law, it is argued that the time for the best of these devices, the mouse, approaches the theoretical minimum. Finally, a Monte Carlo simulation model using gamma-distributed operator times and stochastic method selection rules is described, with which sequences of user actions, time per task, and the distribution of time can be predicted.

The picture of user behavior that emerges from these studies is related to, but distinct from, behavior in classical problem-solving studies. The main difference is that the methods are almost certain of success. For any subproblem the user simply recalls the solution from his experience rather than working it out. Hence there is no search. Such behavior is expected to be found in many cognitive tasks in industrial work and daily life which people perform repetitively, tasks the report calls "routine cognitive skills."

KEY WORDS AND PHRASES

Text editing, user performance, benchmarking, user simulation, cognitive ergonomics, man-machine systems

CR CATEGORIES

4.6, 8.1, 3.36

Contents

Preface and Acknowledgements

The researches reported in this thesis are part of a larger effort called the Applied Information Processing Psychology Project, pursued with Thomas Moran and Allen Newell† at the Xerox Palo Alto Research Center. The general goal of the project is to devise methods, within the general framework of information-processing psychology, for predicting behavior in "applied symbolic tasks"--tasks like writing a business letter, filling in a form, programming a computer, and text editing with a computer.

Each of the six studies reported here is an investigation of how people interact with computer programs for manipulating English text. This topic was the first major investigation by the Project.

It is appropriate to acknowledge the depth of my scientific indebtedness to the various people in the Project for all the studies described here. Since this is a thesis it is also appropriate to be more explicit about my particular contribution than usual. Moran and Newell shared so thoroughly in the ideas and methods on which these investigations are based that they must be considered the product of the three of us. Chapter 2 benefitted from the collection and analysis of data from Editor Y by Marilyn Mantei of the University of Southern California, working as a research intern under my supervision. Teresa Roberts of Stanford University, also working as a research intern, reanalyzed the data to remove error tasks, break it into operators, and count the keystrokes. The chapter is based on internal memoranda by me, Mantei, and two by Roberts. Chapter 3 was improved by suggestions from Newell and Shmuel Oren on sensitivity analysis. Frederick Blackwell from the RAND Corporation ran the experiment. Ivan Sutherland of RAND served as a sort of referee. Chapter 4 was an equal collaboration with Moran and Newell and is based on our report Card, Moran, and Newell (1976). Chapter 5 is likewise an equal collaboration and is based on an earlier unpublished internal report by the three of us. Chapter 6 began as William English's project with me as consultant. English and William Duvall programmed the experiment. When English was transferred to a new assignment, just as subjects were beginning to be run, I took over responsibility for the project. Betty Burr ran the subjects. J. I. Elkind is responsible for major improvements in the design, execution, and presentation of the experiment. E. R. F. W. Crossman, University of California at Berkeley, gave some very helpful suggestions during the planning stage. The comments of Newell,

†Carnegie-Mellon University (consultant for Xerox)

Moran, Roberts, Mantei, and Richard Young† were most helpful in improving the text. A slightly revised form of the chapter will appear in *Ergonomics* (Card, English, and Burr, in press). Chapter 7 draws on two related internal memoranda of Moran's, one on simulating a user and another on the limitations of the model of Chapter 4 for modeling user behavior with Editor Y. Young spent many hours in discussion with me on the chapter. Moran, Newell, and Roberts made useful suggestions on a previous draft. Ronald Kaplan and Beaumont Sheil gave me statistical advice and consed up a wonderful statistical package, IDL, with which to apply it. Any statistical shortcomings are surely the result not of advice they gave me, but of my embarrassment to ask yet more advice when it was needed.

Tracing through the web of this cooperative venture, it will be seen that for all the studies included here I am either the sole, primary, or fully joint author. It is on this basis that I claim this document to properly represent a suitable dissertation.

In all cases, I have adapted the original reports, where they existed, in an attempt to produce a coherent account of the entirety. Thus, and finally, none of the above people are to be held responsible for any errors that have occurred in presenting these scientific ideas or for infelicities of style.

I wish to express appreciation to my committee members Allen Newell, Herbert Simon, and Lee Gregg for their patience in seeing the thesis through and to acknowledge my intellectual debt to them as the originators of many of the information processing concepts applied in the thesis.

It is usual at this point to acknowledge the painful labors of someone in typing and retyping the manuscript. My friends at PARC have, with their inventions, made all of that unnecessary. As a consequence I have been able to produce this document without a typist, using only the help of several PhD computer scientists and a mechanic. I wish to thank Barbara Baird for tuning the laser printer, for piloting the binding machine, for inserting anchored carriage returns in my old PUB files, and for her painful labors in issuing a fair share of the estimated 50,000 editing commands required. I wish to thank William Newman for octal surgery on one of my figures, Dan Swinehart for giving me 12 point Helvetica bold, and Patrick Baudelaire and Joe Maleson for making my splines print.

Finally, it is with fond appreciation that I acknowledge the loving support and impatience of my colleague and wife J.J. She provided advice, editorial review, encouragement, and a necessity for coming to California. For all of these I am grateful.

† University of Edinburgh (visiting scientist at Xerox)

1

Introduction

A person (a "user") sits before a computer terminal with a keyboard for input and a CRT display for output. In the computer is a text file. To the user's left is a manuscript, a printout of the text-file marked with modifications. The user, working through a computer program for text editing, is to effect each of the marked modifications in the text file, producing an updated file. Variations of the task occur with variations in the nature of the computer, the program (the "editor"), the terminal, the size of the manuscript, the type and number of corrections, the physical layout, and the familiarity of the user with the manuscript and the editor.

What can we predict about the behavior of the user? What will be his sequence of actions? How long will it take him to perform them? What sorts of errors will he make?

This thesis presents some answers to these and related questions. Why are they important?

First, the questions are important in their own right. By the 1980s there will be a massive attempt to introduce systems of the sort described above into offices and clerical operations. The well-being of many workers as well as the technical success of the systems themselves will depend on how well the man-computer interface is designed. The quality of the interface will, in turn depend on the knowledge base and theoretical base from which such questions can be answered.

Second, the manuscript editing task is similar to several other tasks. For one thing it is a man-computer task. Increasingly, the industrial world is coming to be populated by tasks of man-computer interaction: electronic cash-register terminals, management information systems, airline reservation systems, "word-processing" systems. For another thing, it is an instance of what might be called a direction-following task: a worker follows simple step-by-step instructions to accomplish a larger end. Examples are a printer following proofreader's marks, a draughtsman implementing corrections indicated on a drawing, a technician building a Heathkit. Detailed study of the manuscript editing task is likely to facilitate work on at least two other large classes of tasks.

Third, the study of the manuscript editing task provides a vehicle for studying a domain of behavior the thesis will call *routine cognitive skills.* The current attempt to understand man as a symbolic information processing system has concentrated on certain domains of behavior: recall and recognition tasks, which reveal the mechanisms of learning and the structure of short-term and long-term memory; discrete symbolic puzzles and mathematical exercises, which reveal the nature of search in problem solving; discrete symbolic induction tasks, which reveal elementary concept acquisition; tasks of elementary sentence comprehension, decision and arithmetic, which reveal the nature of the immediate processor; and simple tasks that occur in child development. There remain, however, important domains of behavior for which we do not yet have any reasonable detailed theory nor any verification that the theory of man as a symbolic information processor provides an appropriate theoretical base. Routine cognitive skills is an example of one such domain.

Fourth, the study of the manuscript editing task helps lay the foundations for an applied information processing psychology. It is not yet quite possible for a psychologist to compute routinely the answers to questions needed by a system designer from a description of the editing task environment the way a bridge designer uses the laws of physics. There is as yet no psychological civil engineering. But there almost is. For elementary single element tasks like reading a pointer dial or reacting to a display of lights it is possible to compute reaction times and order of merit relationships for design alternatives (cf. Welford, 1968) as it is for tracking and feedback display tasks (cf. Poulton, 1974). The various and recent advances in information processing psychology provide some of the methodological and quantitative base from which such a field would draw. The present thesis is the first part of a larger effort directed at making answers to such questions computable.

Background

Despite its practical application and its apparent fruitfulness as a research problem, there have been few studies of manuscript editing. The first studies seem to have been done by Oren (1974, 1975). He derived models for the time to do editing on word-processing machines, a rather different class of systems from the ones studied in this thesis, but did not collect user data or validate the models. Riddle (1976) attempted to evaluate several editors by comparing them with a list of features, but again no observations were made on what users actually do. As far as is known, user behavior for manuscript editing has never been studied.

While manuscript editing has not been studied directly, results from industrial engineering, human-factors, the study of man-machine systems, and psychology provide a context from which the present thesis has proceeded.

Industrial engineers have produced a number of "predetermined time standards" (see, for example, Maynard, 1971). Measurements have been made of clerical tasks (Maynard, Aiken, and Lewis, 1960; Birn, Crossan, and Eastwood, 1961) and even mental operations (Quick, 1962). It is unfortunate that indications of the precision of the measurements, such as their standard deviation, have not been published with them. Furthermore, they are seriously undervalidated—for example, the data supporting the systems is unpublished—a partial consequence of their proprietary nature. What is most satisfying about the industrial engineering studies is the way in which times for novel sequences of operations may be derived from elementary operations. Part of the work in the thesis can be viewed as mental time and motion studies for editing systems.

Whereas industrial engineering studies have tended to concentrate on sequences of operations and to ignore variance in their tabulated measures, human factors studies (for reviews, see Meister, 1976; McCormick, 1976) have tended to do comparisons between static alternatives and to emphasize the analysis of variance. The problems with this literature are the difficulty of extracting numbers on some normalized, natural scale (say "msec/char typed" from a table of "time to type a particular text") to use in predicting new results and the fact that few studies deal with sequences of behavior as opposed to a single operation. One of the most useful digests of human factors results has been the AIR data store (Payne and Altman, 1962) and its successors.

"Man-machine studies" have been developed largely around the problems of aircraft and military operability. This literature, which exists largely in report form, outside of the journals, has been reviewed by

Parsons (1972) and Pew, Baron, Feehrer, and Miller (1977). Interesting models now undergoing development, are the Human Operator Simulator (Strieb, 1975) and SAINT (Seifert, Wortman, and Dukey, 1977).

The older literature of psychology contains a number of classical studies of occupational tasks: Bryan and Harter's (1897, 1899), Tulloss's (1918), and Taylor's (1943) studies of telegraphy, and Book's (1908) studies of typewriting are examples. Regrettably, such studies are today seldom found in the main line psychology journals such as *Psychological Review* and *Cognitive Psychology*. The information-processing psychology literature, however, contains two books from which the general approach of this thesis has been derived: Newell and Simon's *Human Problem Solving* (1972) and Welford's *Fundamentals of Skill* (1968).

Human Problem Solving described problem solving behavior in terms of a person's goals, operators, and methods and how these reflect the structural demands of the problem. Solving a problem is described as a search by the problem solver through a space of knowledge states. The thesis uses these same descriptive elements for the behavior of users in the manuscript editing task. But, although editing is a symbolic task with a well-defined goal, there is no search, no problem space. The selection of methods is completely routine. Yet the task demands considerable cognitive involvement. The analysis of the manuscript editing task is a step toward a generalization of the *Human Problem Solving* theory to include both skill and problem solving behavior.

Fundamentals of Skill reviewed and codified knowledge of skilled performance. The thesis uses parts of the model of human performance and applies its account of hand movement directly. But the main influence of the book has been as an example of how basic and applied research in psychology can be interwoven to the advantage of both. As Professor Welford put the issue in another applied book, *Ageing and Human Skill*:

> It has often been said, with considerable truth, that theoretical studies in psychology divorced from any applied aim quickly lose perspective and become confused with minutiae. On the other hand, it is certain that the applied aim is better served by an understanding of the fundamental changes ... rather than by a series of *ad hoc* studies ... . (Welford, 1958; p1)

The present thesis contains both investigations whose point is basic knowledge and attempted applications of that knowledge.

Measurement strategy

Most of the experiments to be described involve the intensive analysis of a few users performing "naturalistic" tasks. The users are asked to do pretty much what they do in their daily work. For this reason, the laboratory results can be projected to the job with some confidence. But with a small number of subjects how can any claim be made for the reliability of the results?

In order to conduct a scientific inquiry, there must be a source of repetition for the inductive method to work. One way to obtain repetition would be to run many users, making few measurements on each. Another way is to run a few users, but to record their behavior in detail. The reasons for preferring the latter technique in these studies are several:

(1) It is the only way in which to observe the mechanisms which underlie the performance. Suppose there are two methods for performing a task and that 30% of the users do it one way and 70% do it the other way. If the times of the two groups are pooled, there is no easy way in which to understand the number that results. The pooled number represents the behavior of no one. It happens that sometimes users move the text on their terminals to a more convenient position before pointing at the word to be modified and sometimes they do not. Only by detailed observation can the obvious reason for this behavior be understood: they move the text when the target word is in the lower 1/3 of the screen or completely off the bottom.

(2) In many cases, the detailed study of few subjects is much more efficient. There are many ways in which systems can vary. They can use different names for the commands, different commands, different syntax, different terminal speed, response times, screen layouts, text selection schemes. Attempting to perform factorial experiments with several subjects per cell and a few measurements per subject leads to a combinatorial explosion in the number of conditions which must be run. Attempting to run a single factor or a few factors at a time risks missing very important interactions and it is usually easier to comprehend that the user moves the text up when the next target is off the screen than it is to contemplate significance of the Editor-type × Target-distance × Task-type triple interaction. It is often faster to observe a small number of users in detail, so as to understand the mechanisms at work, and then consider how these mechanisms might vary over a population of users.

(3) Even if the above considerations didn't exist, it would still often be difficult to use a large number of users because the total world population of qualified users for a new or experimental system might be two people. Or the effort involved in running an experiment and

analyzing the data for an experimental system might be so large as to make the expense of larger scale experiments unreasonable. The techniques used in this thesis can be seen as one way in which to overcome the often cited "small *N* problem" in industrial man-machine research by making the maximum use of the experimental hours available.

A Typical User-System Dialogue

Before beginning the description of the studies it will be useful to bring the task into focus by describing a typical sequence of operations with an editing system. The user is seated as in Plate 1.1 with the manuscript to one side. She is about to use a typical line-oriented editor called POET (Russell, 1973) to make the corrections indicated in the fragment of manuscript shown in Plate 1.2.

The first marking on the manuscript indicates the word "great" has been mistyped in the manuscript as "geart". A typical dialogue between the system and the user is as follows. (Ellipsis "..." is used to shorten long typeouts by the system; the character # is the "prompt character" the system displays when it is ready for the next command; carriage returns typed by the user are indicated as < CR >; the dialogue proceeds by adding to the bottom line of the display, "scrolling" the rest of the lines up and eventually off the top.)

USER: `"geart"` *(The user indicates the line he wants by typing between quotation marks some characters from the line)*

The system responds internally by making its "Current Line" be the next line it can find that contains the sequence of characters "geart."

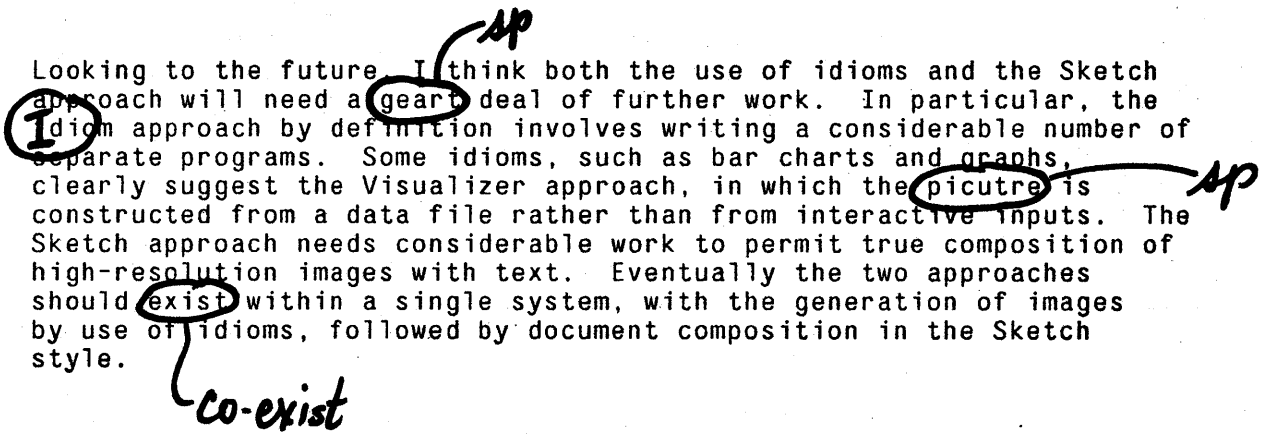
SYSTEM: #
USER: / *(This command requests the system to display its Current Line. The user issues the command to make sure the system is attending to the line he intended.)*

The system displays the Current Line on the bottom line of the display screen.

SYSTEM: `approach will need a geart deal of...`

Now the user decides to make the change by using the system's "substitute" command to substitute the characters "grea" for the character "gear."

Plate 1.1. Physical layout of manuscript editing task.



Looking to the future, I think both the use of idioms and the Sketch approach will need a geart deal of further work. In particular, the idiom approach by definition involves writing a considerable number of separate programs. Some idioms, such as bar charts and graphs, clearly suggest the Visualizer approach, in which the picutre is constructed from a data file rather than from interactive inputs. The Sketch approach needs considerable work to permit true composition of high-resolution images with text. Eventually the two approaches should exist within a single system, with the generation of images by use of idioms, followed by document composition in the Sketch style.

Plate 1.2. Sample fragment of the manuscript used in experiment.

```
USER:    s          (Tells system to use the substitute command)
SYSTEM:  ubstitute  (System completes typing of command name)
USER:    grea<CR>   (Types text to be replaced followed by a carriage return)
SYSTEM:  (for)
USER:    gear<CR>   (Types new text followed by a carriage return)
SYSTEM:  [OK]       (Asks user to confirm that the command is correct before system will execute it)
USER:    <CR>       (Types carriage return to confirm)
```

The system makes the substitution as many times as it can and responds by printing the number of substitutions made—in this case one.

```
SYSTEM: 1
SYSTEM: #
```

Again the user wishes to look at the Current Line to see if, in fact, the modification was made as he intended.

```
USER:   /
SYSTEM: approach will need a great deal of...
SYSTEM: #
```

It was. The user now proceeds to the next task.

In the next task, the word *idiom* is to be capitalized. The user decides he will do this by substituting *Idi* for *idi.* This time he uses the "linefeed method" rather than the "quoted string method" to locate the line to be changed. (Typing the linefeed key is abbreviated <LF>.)

```
USER:    <LF>       (This is the command to print the next line and to make it be the Current Line)
SYSTEM:  idiom approach by definition involves...
SYSTEM:  #
USER:    s
SYSTEM:  ubstitute
USER:    Idi<CR>
SYSTEM:  (for)
USER:    idi<CR>
SYSTEM:  [OK]
USER:    <CR>
SYSTEM:  1
SYSTEM:  #
USER:    /
```

```
SYSTEM: Idiom approach by definition involves...
SYSTEM: #
```

The user proceeds thus through the rest of the manuscript, making the indicated modifications as he comes to them.

The POET editor is typical of a large class of editors in current use design to be usable on a teletype hard-copy terminal. An example of a rather different sort of editor is RCG (a "display" editor inspired by the NLS editor at SRI; see Englebart and English, 1968). This editor uses a five-key chordset by which commands can be entered and a pointing device called a mouse which allows the user to move a cursor on the screen by rolling a small truck across he desk. With this system the user could perform the task as follows.

```
USER:   RC          (Typed on chordset)
SYSTEM: <Displays "Replace Character" at top of
        screen>
USER:   <Points to first char in "idiom" with mouse>
        <Presses button on mouse>
SYSTEM: <Underlines character>
USER:   <Moves hands to keyboard>
        I           (Typed on keyboard)
        <Moves left hand to chordset, right hand to
        mouse>
        <Presses button on mouse>
SYSTEM: <Redisplays entire screen of text with
        change made>
```

The more complex operators POET finds necessary to indicate the target text to be modified are replaced in RCG by a simpler pointing and select operation.

Many other schemes for building an editor are, of course, available. Some will have effects on user performance.

Organization of the Thesis

The six studies described in the following chapters fall into a cyclical pattern of basic research and application. The first basic research study, Chapter 2, seeks to establish by using benchmarks how much difference in user performance there is between different computer editing systems and to learn something of the reasons for the differences. What is learned in Chapter 2 is then applied in Chapter 3 to a prediction problem chosen and refereed by another research institute: when will a computer editing system be faster than a typewriter? Chapter 4 returns

to basic research mode to investigate further the task exhibiting the most interesting differences in the benchmark studies: the manuscript editing task. This time the questions are whether an information processing model can be written for the manuscript editing task and what the appropriate grain size is for the model. Chapter 5 applies the concept of a "unit task" and the grain-size results gained from Chapter 4 to another problem: the prediction of how long it can be expected to take a user to perform various operations on a text processing system as a function of system design. The third time around the basic research-application cycle begins with Chapter 6 and Chapter 7. In Chapter 6, the focus is narrowed to a single sub-operation of the editing process: selecting a target piece of text on the screen of a CRT. In this case established theory from information-processing psychology is able to contribute a well-ordered account of the behavior. In Chapter 7 the empirical results from Chapter 2, the theoretical results of Chapter 4, and the empirical results from Chapter 6 are reduced to a running computer-simulation program for a display editor.

2

A Benchmark Study of Editing Systems

A good place to begin the study of editing is to understand the ranges of performance induced as a consequence of editing system design. The study reported in this chapter examines differences along one of the fundamental dimensions of performance: the speed with which a task can be performed. It does this by examining user performance on several text-editing benchmarks.

Editors

Five editing systems were chosen for study: POET, SOS, TECO, Editor Y, and RCG. These five were chosen because of large differences in their designs and because all could be made to run on local computing equipment. Three of the editors POET, SOS, and TECO are "teletype-type" editors. They operate by printing out one line after another. Two of the editors are "display editors;" they show the user a picture of a page of text and readjust the picture after every editing modification.

POET, a version of QED (Deutsch and Lampson, 1967), is line-oriented but with line numbers which are constantly changing because they specify relative position from the beginning of the file.

SOS (Savitsky, 1969) is a line-oriented editor with fixed line-numbers.

TECO is a character-oriented editor originally written by Daniel L. Murphy at Project MAC, M. I. T. TECO is distinguished by its large number of short commands, its programability, and by the fanaticism it inspires among systems programmers. The experiment used the TENEX TECO version of the editor (Bolt, Beraneck and Newman, 1973).

Editor Y is an experimental display-oriented editor which uses the mouse (English and Englebart, 1967) for pointing at the display. In command structure it is rather similar to POET.

RCG is a display-oriented editor written by William Duvall; it is a descendent of the NLS editor (Englebart and English, 1968). This editor also uses a mouse for pointing and a five-paddle chord device for input of commands.

Benchmark Tasks

The editors were compared by testing users' performances on four benchmark tasks: a Letter Typing, Manuscript Modification, Text Assembly, and Table Typing.

In the Letter Typing task the user typed a letter from a corrected manuscript.

In the Manuscript Modification task the user was given a file containing a memo and a listing of that file with marked modifications. His task was to change the file as specified by the markings.

For the Text Assembly Task the user was to assemble a report by combining three previously written paragraphs stored on separate files with another paragraph to be typed in from a manuscript.

In the Table Typing task, the user typed a simple columnar table with headings.

The materials given to the users for these tasks are reproduced in Appendix 2 at the end of the chapter.

2.1 METHOD

Subjects

Subjects were 10 secretaries and professionals. All were experienced and expert users with the editors on which they were tested. Most users had used the system more than a year and had last used the system within the past week. About a quarter of the users had programmed or

maintained the system.

Design

Because few users were experts in more than one or two of these editors and to avoid the possibility of practice effects from repeated exposure to the tasks, a mixed design was used with the editor as a between subjects variable. Each user was tested on a single editor. Each editor was tested using three users. Three is the smallest number which would give some notion of interuser variability and the largest for which experts on the different editors were available. Only one user was measured using SOS because of its similarity to POET. Each of the four tasks was done with the POET, SOS, TECO, and RCG editors. Performance on Editor Y for the Manuscript Modification task was measured at a later date. As a baseline against which to measure performance, one subject was also measured performing the tasks using an IBM Selectric II typewriter.

Procedure and stimuli

Users were seated in front of a 6 line/sec CRT terminal as shown in Plate 1.1. A session went as follows: The user was first given a set of general instructions urging him to work as fast as possible without making too many errors and stressing that the editor, not the user's abilities was under examination. He was given a warmup task exercising the editor, then each of the four tasks in the order (1) Letter Typing, (2) Manuscript Modification, (3) Text Assembly, (4) Table Typing. The stimulus materials and instructions for each task were bound in a notebook and the subject was allowed to proceed through the tasks at his own pace. The experimental session was recorded on video tape, a video clock superimposing the time to a hundredth of a second on each video frame.

2.2 TIME DIFFERENCES AMONG EDITORS

Table 2.1 shows the time required for each task by users on the different editing systems. Performance on the Letter Typing task mainly reflected the typing abilities of the users. Performance on the Table Typing task mainly reflected the ingenuity of the users: methods varied from typing in the lines directly using fixed tabs provided by the system to making many copies of the first line in the table and substituting for the entries. In the Manuscript Modification task, however, there was a factor of 2.3 difference between the time required by users of the slowest

and the fastest editing system. In the Text Assembly task the ratio between the slowest and fastest system was 1.6. This finding of differences among systems for benchmarks requiring extensive text modifications, but not for benchmarks dominated by typing or idiosyncratic user methods is consonant with other (unpublished) benchmarking studies (Little, Note 1).

TABLE 2.1

TIMES FOR BENCHMARK TASKS (SEC)

Editor	Task			
	Letter Typing	Manuscript Modification	Table Typing	Text Assembly
Typewriter	229	901	483	489
POET	238±28[a]	280±71	244±21	160±65
SOS	315	241	234	147
TECO	252±25	203±42	283±41	131±15
Editor Y	-	133±27	-	-
RCG	224±4	122±16	306±54	102±32

[a] ± indicates standard deviation

It was much faster to use any of the editing systems than to use the typewriter on all tasks except for the Letter Typing task. Since it is assumed modifications are made to the text by retyping it, the time taken by the typewriter depends on the length of the text. But the time required by the editing systems depends on the number of modifications. Hence the ratio of 7.4 between the time to use a typewriter and the time to use the fastest editor would vary with changes in the length of the text and the number of modifications.

For both the Manuscript Modification task and the Text Assembly task, the editors performed in the same order from slowest to fastest: POET, TECO, Editor Y, and RCG. The display-oriented editors, Editor Y and RCG, as a group took about half as long as the scrolling editors to perform the Manuscript Modification task.

The Manuscript Modification task, then, bears closer examination as a place where the design of the editors under review has made a difference in performance time. Table 2.2 displays the performance of the users in greater detail for this task. The manuscript to be edited is a one page letter on which 12 modifications were indicated. Each row in the main part of the table is the time required for one of the users to make the change. Cells marked "-" in the table indicate the user skipped the

TABLE 2.2

TIME PER MODIFICATION FOR MANUSCRIPT MODIFICATION TASK

	POET			SOS	TECO			EDITOR-Y			RCG		
MOD	S4	S6	S13	S12	S18	S19	S20	S16	S30	S31	S12	S14	S15
T1	22.5	35.4	15.4	11.0	11.7	45.5*	10.9	5.9	11.5	19.8	8.4	8.6	8.8
T2	14.5	16.3	7.7	10.1	16.3	9.6	8.8	2.9	4.9	5.2	4.4	6.4*	4.4
T3	24.3*	20.0	9.2	15.6	8.9	9.5	11.6	13.1	11.2	5.6	5.0	6.2	3.9
T4	23.6	59.8*	23.2	29.9*	26.0*	14.1	13.0	8.5	12.0	18.9	21.8	9.5	4.5
T5	16.3	16.5	35.7	19.5*	10.8	18.8*	15.7	7.5	11.4	11.0	5.9	7.7	29.1*
T6	10.6	14.5	20.2	26.5*	7.9	9.3	10.9	10.4	7.6	43.4	5.7	9.0	7.9
T7	18.7	11.4	13.0	23.2*	6.2	8.8	9.1	7.8	11.8	9.4	4.9	5.0	9.8
T8	14.2	14.7	14.0	9.6	29.1*	12.2	12.3	4.8	8.3	14.9*	4.0	4.8	12.6
T9	-	-	8.7	8.2	12.6	11.5	7.8	8.0	-	5.8	7.5	1.9	10.4
T10	-	-	12.0	42.6*	18.6	17.6	19.1	5.4	9.2	10.5	5.4*	-	7.9
T11	10.8	14.3	34.6	8.0	11.3	11.9	5.4	7.4	6.0	-	5.3	3.6	9.5
T12	11.4	13.3	9.2	10.1*	7.6	10.9	5.2	4.6	7.4	9.8	10.5	7.2	7.9
All Tasks			Mean[a]				Mean[a]			Mean[a]			Mean[a]
Mean	16.7	21.6	16.9 (18.5)	17.9	13.9	15.0	10.8 (13.1)	7.2	9.2	14.0 (10.1)	7.4	6.3	9.7 (7.8)
SD	5.3	15.0	9.7 (2.7)	10.8	7.3	10.1	4.0 (2.1)	2.8	2.5	10.9 (3.5)	4.9	2.4	6.6 (1.7)
N	10	10	12 (3)	12	12	12	15 (3)	12	11	11 (3)	12	11	12 (3)
Error-freee Tasks Only													
Mean	15.9	17.4	16.9 (16.7)	10.4	11.2	11.5	10.8 (11.2)	7.2	9.2	13.9 (10.1)	7.5	6.0	8.0 (7.2)
SD	4.9	7.1	9.7 (0.8)	8.8	3.9	2.7	4.0 (0.4)	2.8	2.5	11.5 (3.4)	5.4	2.5	2.7 (1.0)
N	9	9	12 (3)	6	10	10	12 (3)	12	11	10 (3)	10	9	11 (3)

[a] Computed over users, one number per user
* Indicates task on which user made an error

modification. Those modifications on which the user made an error (for example, substituting for the wrong string) are marked with an *. The effect of editing system design on time performance is best seen by examining the error-free "correct modifications." This is not to say that errors are unimportant, only that they require a separate, complementary analysis (with larger quantities of data). A single modification on which an error has occurred can, if the error requires a very long time to correct, seriously distort the editor comparisons. The average time per modification for each user is summarized at the bottom of the table. This calculation is made both for all modifications in the table and again for only those modifications on which the user did not make errors. Overall, the errors increase the time per modification by about 9% (range 0% to 24%). The analyses that follow, therefore, use only the error-free modifications from Table 2.2.

Examining the error-free tasks in Table 2.2 there is still a ratio of 2.3 between the slowest POET and the fastest RCG. This compares with an average ratio of 1.2 between the slowest and fastest user within each editor. Thus, with respect to speed, differences in editor design are considerably more important than differences among expert users.

Since there are small ways in which the fastest editor RCG could be improved and editors are known to exist which are even slower than POET, it is probably justified to say, as a rough statement, that the design of an editor makes a factor of 3 difference in the time to make typical modifications to a manuscript a manuscript.

2.3 SOURCES OF THE TIME DIFFERENCES

What is the source of the observed differences in the time to use the editors?

One way to look at the differences among editors is to consider how much work the user has to do in order to accomplish a modification to the text and one index of work is the number of keystrokes required. In Figure 2.1 the time per modification is plotted against the keystrokes per modification (for the user with the lowest error rates in each editor: S4, S18, S30, S14). POET, SOS, TECO, and RCG fall exactly on a line essentially through the origin (T_{mod} = (0.26) + 0.57 $N_{keystrokes}$ sec; R^2 > .999, s_e = 0.12 sec). Editor Y, however, takes four seconds per modification (about a factor of 2) longer than expected. It is possible that counting keystrokes does not capture some important part of the interaction. More detailed comparison of the behavior of users using Editor Y suggests that the users spent more time than expected in getting

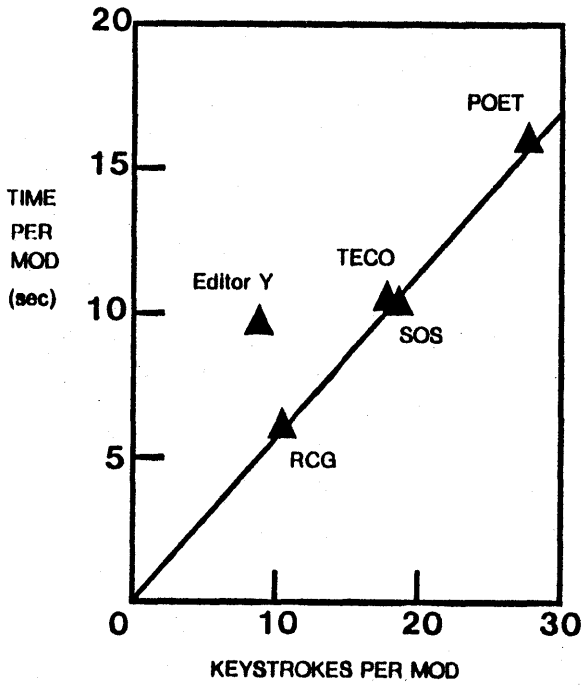


Figure 2.1. Mean time per modification for editors as a functiion of the keystrokes per modification

the task with Editor Y and that the time required by the numerous pointing operations needs to be considered. A more definitive explanation requires additional experimentation.

2.4 CONCLUSION

The design of an editor makes roughly a factor of 3 difference in the time to edit a manuscript.

The number of keystrokes which must be typed to effect a modification is an important factor in the time it takes to perform the modification.

An approximation to the time per modification can be computed by multiplying the number of keystrokes by 0.6 sec/keystroke.

Of course, as with any such simple method, some caution is in order. There may be other features of the system, as in the case of Editor Y, which have to be taken into account. Very brief command sequences may also be harder to remember and thus increase substantially system training time.

REFERENCE NOTE

1. Little, L. Personal communication on word processing benchmark studies conducted by him at Lawrence Livermore Laboratories.

APPENDIX: BENCHMARK TASKS

The following pages contain the instructions and material to be edited which was given to the users in this experiment.

GENERAL INSTRUCTIONS

Please read these instructions to yourself as I read them aloud.

The purpose of this experiment is to assess the ease with which certain common tasks can be performed with the current editor. It is not a test of your abilities. It is a test of the editor.

You should try to perform the following tasks at your usual working speed. That is, I want you to work as swiftly as you can without making many mistakes. Should you happen to make a mistake, simply correct it as quickly as you can and go on.

If it is natural for you to do so, and if you can do it without giving it any thought, you are encouraged to talk aloud as you work, saying what you are doing. However, do not "explain" or "justify" your actions to me and do not "introspect" on what you think you are doing as these things take time and will interfere with the tasks you are doing.

Any questions?

INSTRUCTIONS FOR TASK 1

The next page is the text of a letter on which some corrections have been marked. Your task is to type a new copy of the letter and to save it on a file called LET1. You need not indent the precise number of columns present in the text so long as the appearance is similar.

July 7, 1975

Manager, Credit Department
Johnstone's Department Store

Gentlemen:

On April 7 Miss Heather Smith, account number 86153, returned for credit a clock radio that she purchased from you on April 2.

Miss Smith's July statement does not show this credit. She now has in her possession the credit slip the clerk gave her at the time the radio was returned. Miss Smith would appreciate it if you would verify the credit and send her a corrected statement. The price was $57.20, tax included.

In the meantime, please find Miss Smith's check for ~~$420.51~~ 435.61, which is the amount of the statement less the price of the returned merchandise.

Sincerely yours,

Benjamin M. Ink

INSTRUCTIONS FOR TASK 2

On the next page is the text of a memo on which several corrections have been indicated. This memo has been stored in your machine under the name MEM1. You are to use your editor to make the indicated changes, storing the corrected memo under the name MEM1A.

To: John Ingram (Northeast) Date: June 22, 1976
Betty Grailing (Southern)

From: Dwaard Finwald

Subject: ~~Datatran~~ DATATRAN Sales Activity Promotion

The attached information ~~represents~~ is a DATATRAN promotion proposed by the DATATRAN Business Center.

The program represents a sales activity driven strategy to attain the DATATRAN operating plan for 1976, without detracting from ~~our~~ other products demands.

The program is comprised of two phases:

1. Phase I affects all salesmen including Account Managers and ASR's Sales Managers and BSTM's, and it directed to DATATRAN sales order activity.

2. Phase II is aimed at attaining net installations/net add plan activity for the third quarter for DATATRAN and is specifically geared toward the branch manager, branch sales manager/branch sales planning manager, and ~~Production~~ Customer Engineers.

The program will be administered by Field Operations in conjunction with the A. C. Abercrombie Company.

The program has been reviewed and improved in concept by Sherwood Anderson and Elwood Reisling, and is reported to you for your review and comments.

The program is planned for implementation May 1st, with Phase I ending August ~~30~~ 25th, while Phase II will be extended through the end of October.

Individual awards and qualification levels are contained in the attached package.

I would appreciate your review of this material and comments/critique prior to Friday, March ~~18~~ 4.

In the event that other Business Center or individual product groups plan a promotion during ~~a similar~~ the same period, the attached promotion will be adjusted to reflect a balance performance for attainment on behalf of the sales managers/branch ~~sales managers branch~~ managers.

In that connection, I ~~would~~ will keep you apprised of any additional developments.

Thank you for your assistance, and I ~~believe~~ am sure that activity of this nature is necessary to ensure a successful third quarter for DATATRAN devices as well as all products.

DF/br

INSTRUCTIONS FOR TASK 3

In this task you are first to type in the text on the next page, then assemble the rest of the manuscript from paragraphs pre-stored on your machine. You will also have to add a blank line between the paragraphs. The paragraphs to be assembled are stored as files

TICS
IDPS
POGOS

and they should be assembled together on the file OUT in that order.

During the past six months, improvement of the systems software has continued and software for a second phase document processing/communications has been specified and programming begun. Major accomplishments include:

INSTRUCTIONS FOR TASK 4

On the next page is a table. Please type the heading and indicate portions of the table and file it under TAB1.

322 Non-Parametric Statistics

CRITICAL VALUES FOR THE KOLMOGOROV-SMIRNOV TEST OF GOODNESS OF FIT

Sample Size (n)	Significance Level .20	.15	.10	.05	.01
1	.900	.925	.950	.975	.995
2	.684	.726	.776	.842	.929
3	.565	.597	.642	.708	.829
4	.494	.525	.564	.624	.734
5	.446	.474	.510	.563	.669
6	.410	.436	.470	.521	.618
7	.381	.405	.438	.486	.577
8	.358	.381	.411	.457	.543
9	.339	.360	.388	.432	.514
10	.322	.342	.368	.409	.486
11	.307	.326	.352	.391	.468
12	.295	.313	.338	.375	.450
13	.284	.302	.325	.361	.433
14	.274	.292	.314	.349	.418
15	.266	.283	.304	.338	.404
16	.258	.274	.295	.328	.391
17	.250	.266	.286	.318	.380
18	.244	.259	.278	.309	.370
19	.237	.252	.272	.301	.361
20	.231	.246	.264	.294	.352
25	.21	.22	.24	.264	.32
30	.19	.20	.22	.242	.29
35	.18	.19	.21	.23	.27
40				.21	.25
50				.19	.23
60				.17	.21
70				.16	.19
80				.15	.18
90				.14	
100				.14	
Asymptotic Formula:	$\frac{1.07}{\sqrt{n}}$	$\frac{1.14}{\sqrt{n}}$	$\frac{1.22}{\sqrt{n}}$	$\frac{1.36}{\sqrt{n}}$	$\frac{1.63}{\sqrt{n}}$

Reject the hypothetical distribution $F(x)$ if $D_n = \max |F_n(x) - F(x)|$ exceeds the tabulated value. (For $\alpha = .01$ and $.05$, asymptotic formulas give values which are too high—by 1.5 per cent for $n = 80$.)

3

APPLICATION: Predicting When an Editing System is Better Than a Typewriter

In the spring of 1975 the Applied Information Processing Psychology Project was challenged by Ivan Sutherland and Frederick Blackwell at the RAND Corporation in Santa Monica to predict, in advance, the results of an experiment to be performed there.

It is sometimes said that because it takes less time to set up a typewriter than a computer text editor, the former is better for short jobs and the latter for long jobs. The basic purpose of their experiment was to find the "crossover point," the length of text where the computer's speed in making corrections began to outweight the typewriter's ease of setup.

This chapter presents the model which arose in response to this challenge. The challenge provided the first opportunity to apply the results of Chapter 2 to the prediction of user performance.

3.1 MODEL

Brief Description of the RAND Experiment

The experiment compared the time required to make modifications to five texts of varying lengths using an electric typewriter with the time to make the modifications using the WYLBUR editing system (Stanford, 1975) running on a time-shared computer. The subjects were 12 professional secretaries, each a proficient user of WYLBUR and of a typewriter. Each user edited all five texts twice, once with the typewriter, once with WYLBUR. Half the users used the typewriter first, half the editor. The order in which the texts were edited was varied systematically.

Basic Typing and Editing Models

The time T_t to produce a new copy of the same manuscript using a typewriter depends only on the length of the manuscript and the setup time of the typewriter.

$$T_t = T_{st} + LT_l \tag{3.1}$$

where

T_{st} = Time to set up typewriter (in sec)
L = Length of the text (in lines).
T_l = Time to type a line (sec)

The time T_e to perform a manuscript editing task, on the other hand, depends on the number of modifications. Suppose that every modification with an editing system took the same amount of time T_u to perform. Suppose furthermore that secondary effects such as operator

fatigue and time to turn the page were absent or negligible. Then the T_e would be given by

$$T_e = T_{se} + N_u T_u \tag{3.2}$$

where

T_{se} = Time to set up the editor (in sec)
N_u = Number of modifications ("unit tasks") to be made.
T_u = Time to do one editing modification (in sec).

Expressing Equation 3.2 in terms of the modification density, $\rho = N_u/L$, makes it more comparable to Equation 3.1:

$$T_e = T_{se} + \rho L T_u. \tag{3.3}$$

Length Crossover Point L_c

If the typewriter is faster to set up ($T_{st} < T_{se}$), but the editor is faster for making modifications ($\rho T_u < T_l$), then there exists some document length L_c called the length crossover point such that for

$L > L_c$ the editor is faster;
$L < L_c$ the typewriter is faster.

To find L_c we use Equation 3.2 and Equation 3.3. The time for the editor and the typewriter will be the same when

$$T_{se} + \rho L_c T_u = T_{st} + L_c T_l.$$

That is,

$$L_c = (T_{se} - T_{st})/(T_l - \rho T_u). \tag{3.4}$$

Thus, for

$L > (T_{se} - T_{st})/(T_l - \rho T_u)$, the editor is faster; for
$L < (T_{se} - T_{st})/(T_l - \rho T_u)$, the typewriter is faster.

Density Crossover Point ρ_c

Similarly there exists a certain density ρ_c called the *density crossover point* such that for

$\rho < \rho_c$, the editor is faster; for
$\rho > \rho_c$, the typewriter is faster.

Solving Equation 3.4 for ρ gives

$$\rho_c = T_l/T_u - (T_{se} - T_{st})/LT_u \tag{3.5}$$

Prediction of L_c *for RAND experiment*

What prediction can be made for the RAND editing experiment? First we need to estimate the parameters of the above equations as best we can.

T_{st} In connection with the experiment of Chapter 2 several measurements were avialable for the setup times for a typewriter: 20 sec, 32 sec, 19 sec (mean 23.7 sec), which rounds to

$$T_{st} \doteq 24 \text{ sec.}$$

T_{se} Also in connection with the experiment of Chapter 2 measurements were collected of the setup time for POET and SOS editors (both of which resemble WYLBUR): 14 sec, 16 sec, 13 sec, 5 sec, 13 sec (mean 12.2 sec). Add to that about 25 sec to log into the computer (measured time to telephone a local computer and log into the TENEX operating system).

$$T_{se} \doteq 37 \text{ sec}$$

ρ Measurements of the five texts used in the RAND experiment yielded the values of ρ listed in Table 3.1. The individual tasks varied from $\rho = 0.38$ to $\rho = 0.64$ with an average of

$$\rho \doteq 0.58 \text{ mod/line.}$$

T_u Measurements made in connection with the experiment in Chapter 2 on the POET and SOS editors gave an average time per modification of

TABLE 3.1

MEASURED PARAMETERS FOR TEXTS USED IN EXPERIMENT

		Text					
		T1	T2	T3	T4	T5	All
L	(lines)	4	10	21	26	90	151
N_u	(mods)	2	6	8	14	58	88
ρ	(mods/line)	0.50	0.60	0.38	0.54	0.64	0.58

$$T_u \doteq 20 \text{ sec}^1.$$

T_l The average typing rate for POET users in the last chapter was 0.22 sec/char. Since there are 63 char/line,

$$T_l \doteq 14 \text{ sec/line}$$

Notice that in the time it takes to make one correction with WYLBUR, the user could have typed $L = T_u/T_l$ = (20 sec)/(14 sec/line) = 1.4 lines = 16 words. It is evident that, contrary to popular opinion, it is much more prudent to type more slowly but carefully on an editing system than it is to type at high speed and correct the errors later.

The length crossover point L_c is predicted to be

$$\begin{aligned} L_c &= (T_{se} - T_{st})/(T_l - \rho T_u) \\ &= (37 - 24)/(14 - 0.58 \times 20) \\ &= 5.4 \text{ lines.} \end{aligned}$$

The density crossover point ρ_c is predicted to be

$$\begin{aligned} \rho_c &= (T_l/T_u) - (T_{se} - T_{st})/LT_u \\ &= 0.70 - 0.65/L. \end{aligned}$$

As $L \rightarrow \infty$, $\rho_c \rightarrow 0.7$ modifications/line. Another way of puting this result is: if there is more than one modification to be done every $(0.7)^{-1} = 1.4$ lines, then it is better to retype the text from scratch.

Plotting the time to modify a text predicted by Equation 3.2 and Equation 3.3 as a function of the length of the text, it is apparent (Figure 3.1) that the editor beats the typewriter immediately for almost any task but addressing envelopes. But Figure 3.1 reveals that as the length of the text increases the editor does not continue to increase its superiority as much as might be expected (the bump in the WYLBUR curve comes from the low density for task T3). Why?

The answer is that the density $\rho = 0.58$ chosen for the experiment just happens to be near the critical crossover density ρ_c! Had the experiment varied ρ, one text at the critical value would not have been a problem, but as each of the texts sits near this critical point, local fluctuations in T_u or ρ will push the value of $T_l - \rho T_u$ back and forth. Another way to display the model's prediction is to plot the density crossover point ρ_c as a function of text length L (Figure 3.2) using Equation 3.5. Note how near the tasks are to the crossover density line. *Because all of the texts sit near the density crossover line, it can be predicted that the results of the RAND experiment will be equivocal: the length crossover point* L_c *will not be well-defined.*

What about predictions at other values of ρ? The prediction of task time as a function of length of text for different values of ρ is plotted in Figure 3.3. The typewriter either wins or loses immediately. This is true because the difference in time required to set up for a typewriter and to set up for WYLBUR is (for task lengths > 5 lines) a small percentage of time required to do the task.

3.2 CONFIDENCE INTERVALS

Recall that the analysis so far is entirely a prediction for the RAND experiment. Only the stimulus materials were available. All other parameters, including the typing rates of the users, were taken from other pre-existing experiments by analogy. Thus the parameters from which these predictions were produced are uncertainly determined. To what extent are the conclusions dependent on the goodness of the parameter estimates? One way to determine the effect of this uncertainty is to compute some sort of confidence intervals for the the predictions.

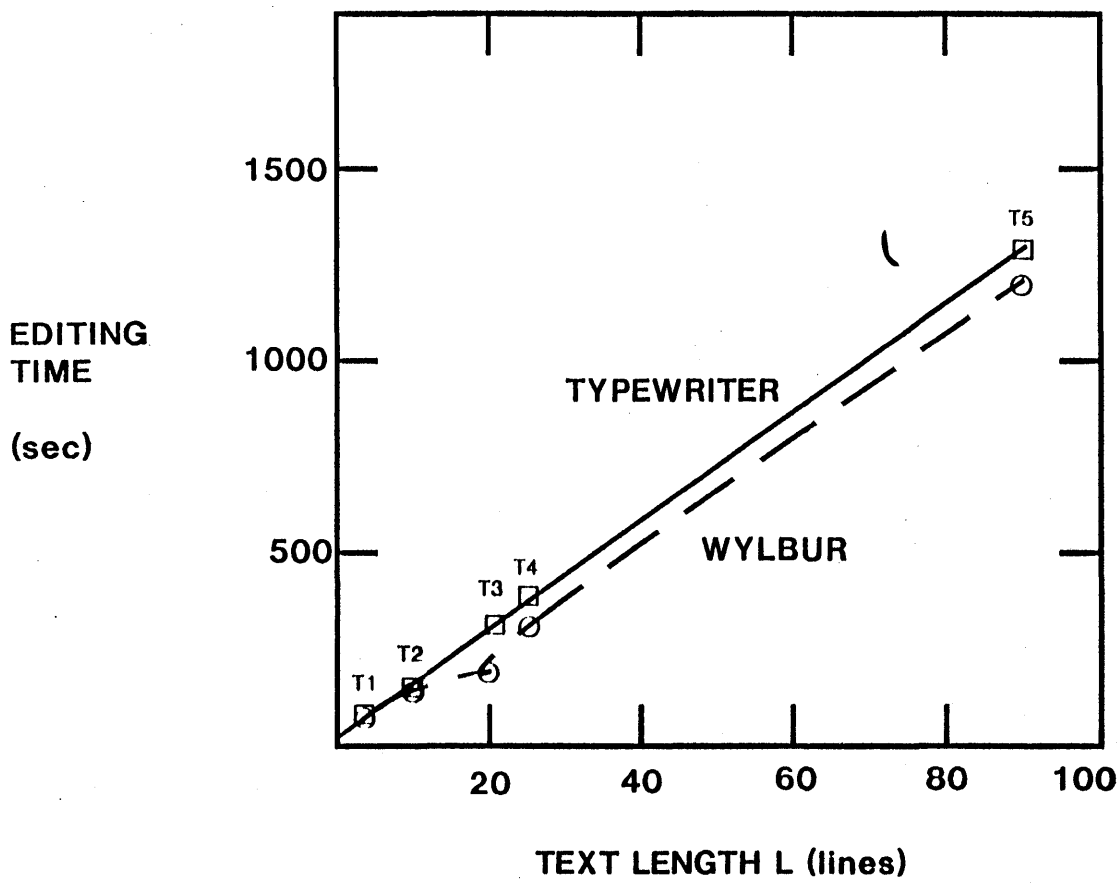


Figure 3.1. Prediction of editing and typing times for tasks

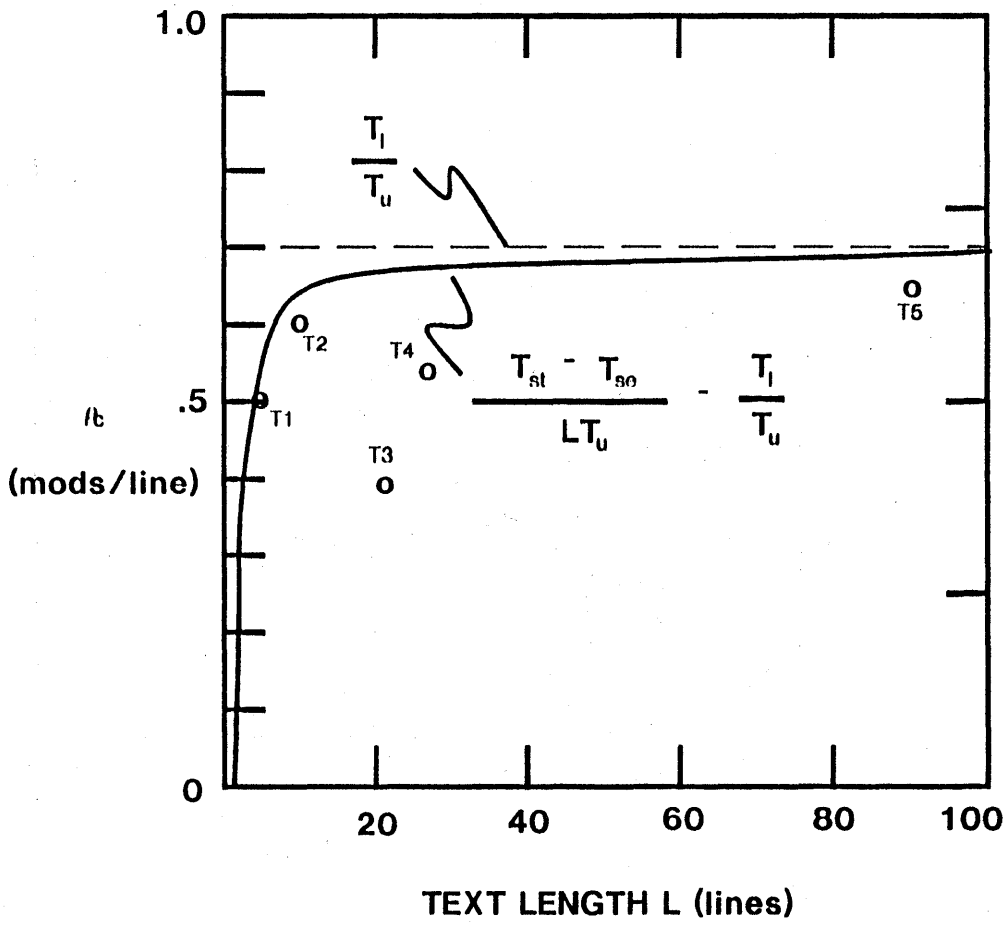


Figure 3.2. Density Crossover Point ρ_c as a function of Text Length L

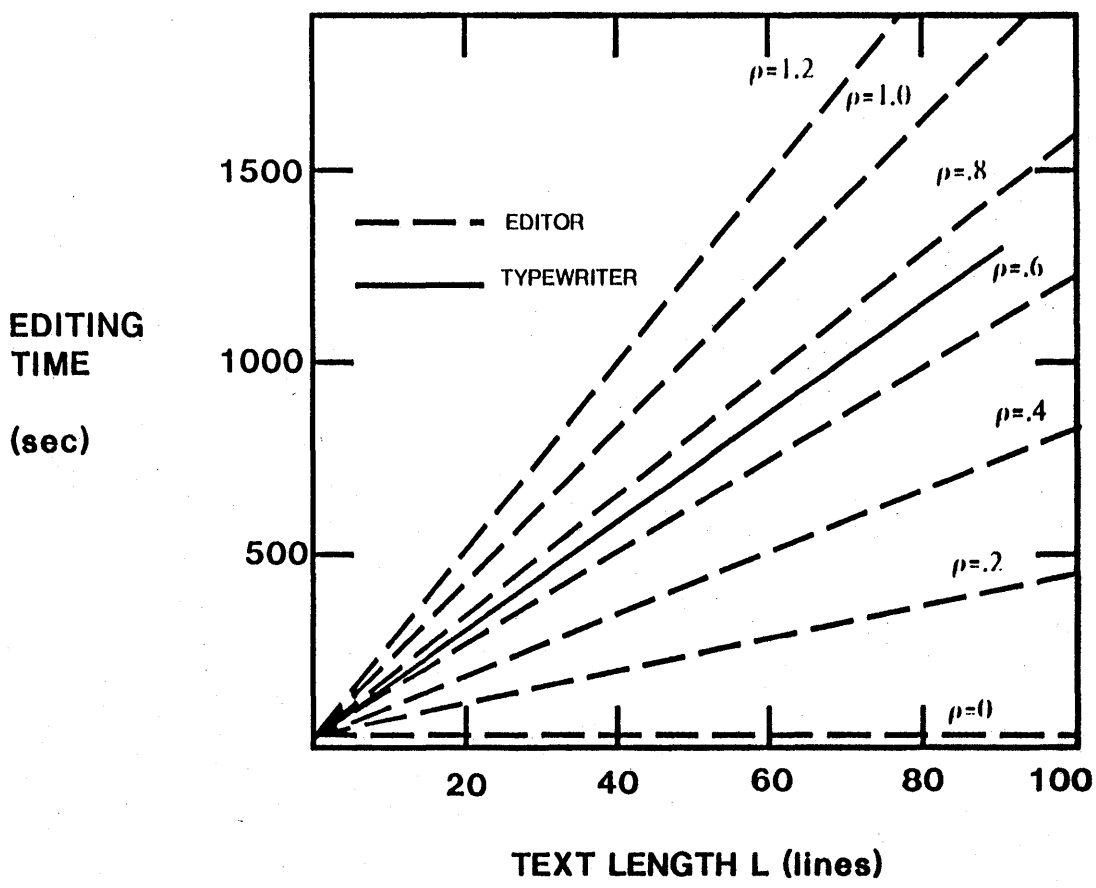


Figure 3.3. Prediction of task times at different modification densities ρ (modifications/line)

TABLE 3.2

ASSUMED PROBABILITY DISTRIBUTION
FOR ESTIMATING CONFIDENCE INTERVALS

	Probability		
	0.15	0.70	0.15
T_{st} (sec)	15	24	40
T_l (sec/line)	10	14	20
T_{se} (sec)	3	37	60
T_u (sec/mod)	15	20	40

Confidence Intervals for L_c

While we do not have a formal probability distribution for the likely values of the parameters, we might claim to have *some* knowledge about the distrubution. For example, from observing a few people set up a typewriter, from examining the standard deviation of the times measured for this activity, it can be concluded with some confidence that the probability that $T_{st} \geq 10^6$ sec is vanishingly small. In Table 3.2 a high value and a low value has been estimated for each parameter and each given a probability of 0.15. From these estimated probability density functions it is possible to compute the probability density functions for Equation 3.2 to Equation 3.5.

The results of this rather lengthy computation are plotted in Figure 3.4. For texts T3, T4, and T5, WYLBUR should surely be faster. But for T1 and T2 it is more difficult to predict.

Confidence Intervals for ρ_c

From a similar computation, Figure 3.5 plots confidence intervals for ρ_c. The figure shows there is quite some uncertainty in the asymptote for the curve. This comes about because the asymptote is given by T_l/T_u. If the numerator were to get larger by a factor of two and the denominator smaller by a factor of two it would throw the asymptote off by a factor of four. The curve also illustrates that the transient part of

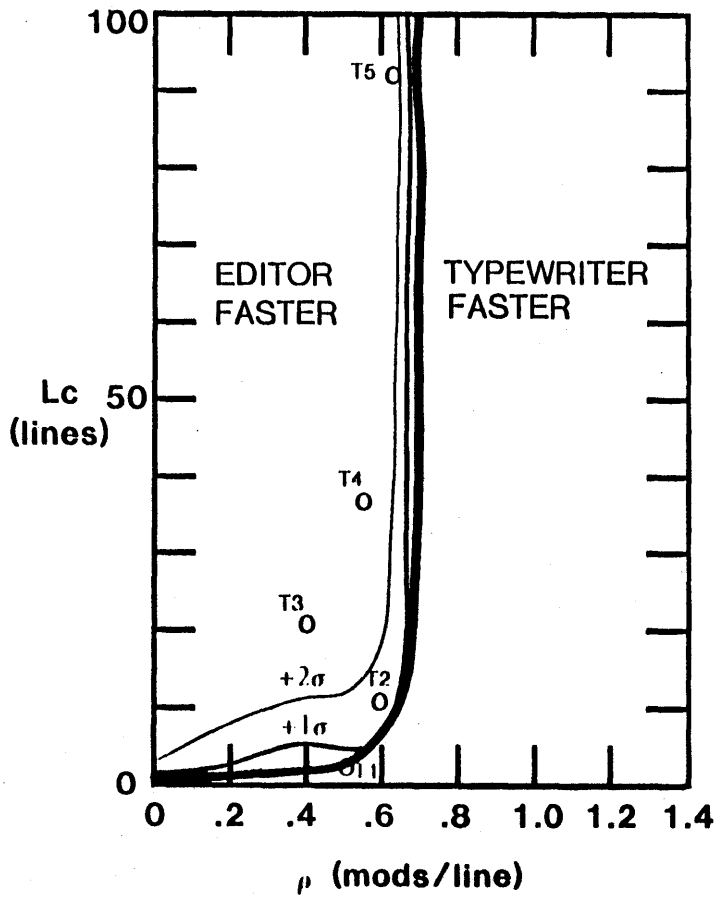


Figure 3.4. Confidence intervals for L_c

Equation 3.5 is of negligible consequence for practical purposes since its influence is small for texts larger than 10 lines. In fact its influence is quite modest for texts larger than 5 lines. All of the texts lie in the $\pm 1\sigma$ confidence band. Thus predictions of which side of the critical density line the texts lie on are very uncertain.

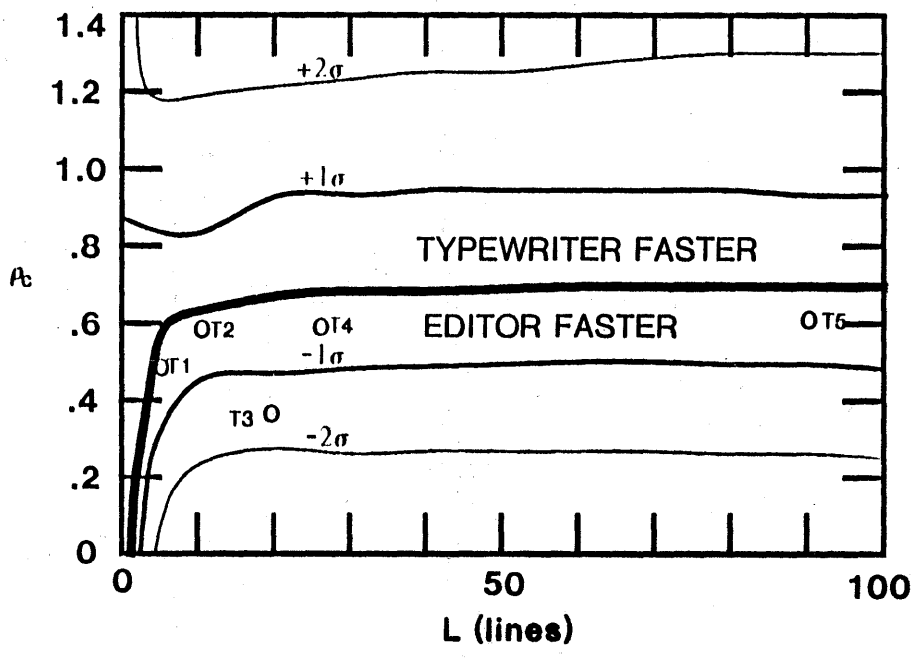


Figure 3.5. Confidence intervals for ρ_c

3.3 SENSITIVITY ANALYSIS

Another way in which to determine the effect of uncertainties in the parameters is to see how sensitive the predictions of the equations are to jiggles in the parameter values.

Length Crossover Density (Equation 3.4)

Let A be the point $\langle \rho, T_u, T_{se}, T_{st}, T_e \rangle = \langle 0.58$ mod/line, 20 sec, 24 sec, 37 sec, 14 sec $\rangle$ in the space of all possible values of the parameters in Equation 3.4. Let B = $\langle \rho', T_u', T_{se}', T_{st}', \text{Te}' \rangle$ be some other point in that space. Using a Taylor expansion about A, L_c at B can be approximated by

$$L_c(\mathrm{B}) \simeq L_c(\mathrm{A}) + ((\mathrm{B} - \mathrm{A}) \cdot \nabla) L_c(\mathrm{B})$$

where ∇ is the gradient operator which indicates that partial derivaties should be computed with respect to each of the components of **B**.

Writing L_c' as shorthand for $L_c(\mathbf{B})$ and just L_c for $L_c(\mathbf{A})$,

$$\begin{aligned} L_c' &\simeq L_c + [(\rho' - \rho)\partial/\partial\rho + (T_u' - T_u)\partial/\partial T_u \\ &\quad + (T_{se}' - T_{se})\partial/\partial T_{se} + (T_{st}' - T_{st})\partial/\partial T_{st} \\ &\quad + (T_l' - T_l)\partial/\partial T_l]\, L_c \\ &= L_c + [\partial L_c/\partial\rho]\,(\rho' - \rho) \\ &\quad + [\partial L_c/\partial T_u]\,(T_u' - T_u) \\ &\quad + [\partial L_c/\partial T_{se}]\,(T_{se}' - T_{se}) \\ &\quad + [\partial L_c/\partial T_{st}]\,(T_{st}' - T_{st}) \\ &\quad + [\partial L_c/\partial T_l]\,(T_l' - T_l). \end{aligned}$$

In order to normalize the magnitudes of the coefficients and the results, we express this equation in a ratio form:

$$\begin{aligned} &(L_c' - L_c)/L_c \\ &\quad \simeq [\rho/L_c\ \partial L_c/\partial\rho]\,(\rho' - \rho)/\rho \\ &\quad\quad + [T_u/L_c\ \partial L_c/\partial T_u]\,(T_u' - T_u)/T_u \\ &\quad\quad + [T_{se}/L_c\ \partial L_c/\partial T_{se}]\,(T_{se}' - T_{se})/T_{se} \\ &\quad\quad + [T_{st}/L_c\ \partial L_c/\partial T_{st}]\,(T_{st}' - T_{st})/T_{st} \\ &\quad\quad + [T_l/L_c\ \partial L_c/\partial\rho]\,(T_l' - T_l)/T_l \end{aligned}$$

or, using δx for $(x' - x)/x$,

$$\begin{aligned} \delta L_c &\simeq [\rho/L_c\ \partial L_c/\partial\rho]\,\delta\rho + [T_u/L_c\ \partial L_c/\partial T_u]\,\delta T_u \\ &\quad + [T_{se}/L_c\ \partial L_c/\partial T_{se}]\,\delta T_{se} + [T_{st}/L_c\ \partial L_c/\partial T_{st}]\delta T_{st} \\ &\quad + [T_l/L_c\ \partial L_c/\partial T_l]\,\delta T_l \end{aligned}$$

Evaluating the derivatives and substituting $(T_{se} - T_{st})/(T_l - \rho T_u)$ for L_c gives

$$\begin{aligned} \delta L_c &= (T_l/\rho T_u - 1)^{-1}\,(\delta\rho + \delta T_u) + (1 - T_{st}/T_{se})^{-1}\,\delta T_{se} \\ &\quad + (1 - T_{se}/T_{st})^{-1}\,\delta T_{st} + (\rho T_u/T_l - 1)^{-1}\,\delta T_l \qquad (3.6) \end{aligned}$$

Equation 3.6 expresses relative changes in L_c as a linear combination of relative changes in the parameters of Equation 3.4. The percentage change in L_c is approximated as the sum of the percentage changes due to each variable. The relative sensitivity of predicted L_c due to the different parameters may thus be assessed directly from the relative size of the coefficients. At $\rho = 0.6$ Equation 3.6 becomes

$$\delta L_c = 6.00\ \delta\rho + 6.00\ \delta T_u + 2.85\ \delta T_{se} - 1.85\ \delta T_{st} - 7.00\ \delta T_l$$

A 1% error in T_l will produce a 7% error in L_c. The values of the coefficients for other values of ρ are plotted in Figure 3.6, as are those of the three following equations. The value of L_c is more sensitive to changes in T_u, ρ, and T_l than to changes in T_{se} and T_{st}, the ostensible parameters of interest. The sensitivity analysis makes it quite clear (1) that the prediction of $L_c = 5$ lines from the model is not robust over changes in the parameters and that (2) it will be difficult to maintain adequate control over the variables in the experiment at this level of ρ. Considerable variance in the measured value of L_c is predicted. The figure shows that the coefficients for $\delta\rho$, δT_u, and δT_l are all very large in the region between $\rho = 0.06$ and $\rho = 0.08$. Conversely, had the experiment chosen $\rho = 0.2$, then

$$\delta L_c = 0.40\ \delta\rho + 0.40\ \delta T_u + 2.85\ \delta T_{se} - 1.85\ \delta T_{st} - 1.40\ \delta T_l$$

In this case L_c would have been much less affected by the parameters other than T_{se} and T_{st}

Density Crossover Point ρ_c (*Equation 3.5*)

Proceding similarly for Equation 3.5,

$$\begin{aligned}\delta\rho_c \simeq\ & [T_{st}/\rho_c\ \partial\rho_c/\partial T_{st}]\ \delta T_{st} + [T_{se}/\rho_c\ \partial\rho_c/\partial T_{se}]\ \delta T_{se} \\ & + [L/\rho_c\ \partial\rho_c/\partial L]\ \delta L + [T_u/\rho_c\ \partial\rho_c/\partial T_u]\ \delta T_u \\ & + [T_l/\rho_c\ \partial\rho_c/\partial T_l]\ \delta T_l \\ =\ & [1 - (T_{se} - LT_l)/T_{st}]^{-1}\ \delta T_{st} \\ & + [(T_{st} - LT_l)/T_{se} - 1]^{-1}\ \delta T_{se} \\ & + [LT_l/(T_{se} - T_{st}) - 1]^{-1}\ \delta L - \delta T_u \\ & + [1 - (T_{se} - T_{st})/LT_l]^{-1}\ \delta T_l \end{aligned} \qquad (3.7)$$

At $L = 20$ lines, Equation 3.7 becomes

$$\delta\rho_c = 0.09\ \delta T_{st} - 0.13\ \delta T_{se} + 0.05\ \delta L - 1.00\ \delta T_u + 1.05\ \delta T_l$$

A 1% change in either T_u or T_l will produce a 1% change in ρ_c. A 1% change in the other parameters produces only a negligible change in ρ_c. For texts of reasonable length (> 10 lines), ρ_c will depend mainly

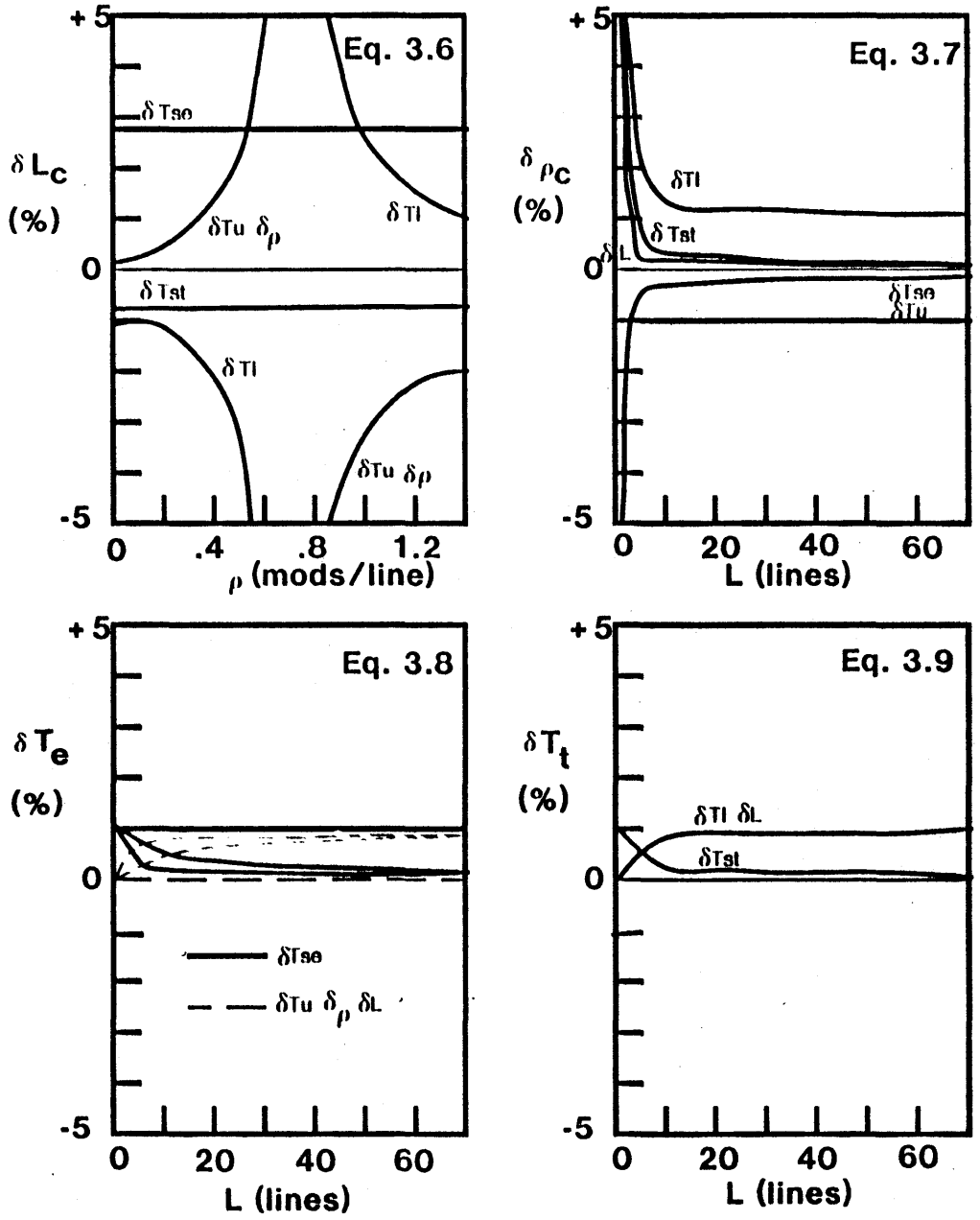


Figure 3.6. Values of coefficients in sensitivity equations

on T_u and T_l.

Total Editing Time T_e (Equation 3.1)

$$\begin{aligned}\delta T_e \simeq\ & [T_{se}/T_e\ \partial T_e/\partial T_{se}]\ \delta T_{se} + [\rho/T_e\ \partial T_e/\partial \rho]\ \delta \rho \\ & + [T_u/T_e\ \partial T_e/\partial T_u]\ \delta T_u + [L/T_e\ \partial T_e/\partial L]\ \delta L \\ =\ & (1 + \rho L T_u/T_{se})^{-1}\ \delta T_{se} \\ & + (1 + T_{se}/\rho L T_u)^{-1}\ (\delta \rho + \delta T_u + \delta L) \qquad (3.8)\end{aligned}$$

At $L = 20$ lines, $\rho = 0.6$ mod/line,

$$\delta T_e = 0.13\ \delta T_{se} + 0.87\ \delta \rho + 0.87\ \delta T_u + 0.87\ \delta L.$$

Sensitivity to T_{se} fades quickly as L increases. A 1% change in the other arguments produces a little less than a 1% change in T_e.

Total Typing Time T_t *(Equation 3.2)*

$$\begin{aligned}\delta T_t \simeq\ & [T_{st}/T_t\ \partial T_t/\partial T_{st}]\ \delta T_{st} + [L/T_t\ \partial T_t/\partial L]\ \delta L \\ & + [T_l/T_t\ \partial T_t/\partial T]\ \delta T_l \\ =\ & (1 + L T_l/T_{st})^{-1}\ \delta T_{st} \\ & + (1 + T_{st}/L T_l)^{-1}\ (\delta L + \delta T_l) \qquad (3.9)\end{aligned}$$

At $L = 20$ lines,

$$\delta T_t = 0.08\ \delta T_{st} + 0.92\ \delta L + 0.92\ \delta T_l.$$

Again sensitivity to T_{st} the setup time fades quickly as L increases. And again a 1% change in the other parameters produces a little less than a 1% change in T_t.

What the results of the confidence interval and sensitity analyses tells us is that while it may be possible to predict the value of L_c functionally, that is, to produce an equation whose evaluation will give a reasonable value for L_c, it is not possible to predict the value of L_c numerically with any certainty on this set of texts because they are all set so near to ρ_c. Small errors in the parameter values will cause large errors in the predictions. The analyses tell us further that the experiment is not likely to produce a well defined value of ρ_c against which to compare a prediction. On the other hand, predictions of total time to process each text are likely to be reasonable and, in fact, to depend very little on the setup times of the editor or the typewriter.

3.4 RESULTS AND DISCUSSION

By agreement the predictions above were mailed to RAND at the same time in which they mailed the results from the experiment. At the time at which the exchange took place, the data from only 8 subjects had been tabulated. The following analyses are based on the results of those 8 subjects.

Figure 3.7 shows the time for each task as a function of the length of the task for both the typewriter and for WYLBUR and is comparable to the prediction in Figure 3.1. As predicted, the crossover point L_c was not well-defined. Connecting the mean observed times produces three crossover points. The heavy overlap of the error bars makes it unlikely that the times for texts T2, T3, and T5 are reliably different from one another.

TABLE 3.3

PARAMETER ESTIMATES

		Pred.	Obs.	%Dif
T_{st}	(sec)	24	5	-85%
T_l	(sec/line)	14	18	22%
T_{se}	(sec)	37	179	649%
T_u	(sec/mod)	20	16	-20%

How good were the simple models of typing and editing in Equation 3.2 and Equation 3.3? The comparison needs to be made in two ways. First, how good were the models at predicting the result in advance of any knowledge about the outcome? This *zero-parameter prediction* is usual in practice where, as in this case, good values for the parameters are not known. Second, how good were the models at predicting the result given knowledge of the parameter values? This *two-parameter prediction* (two values must be estimated from the data) allows an evaluation to be made of the accuracy of the functional form of the

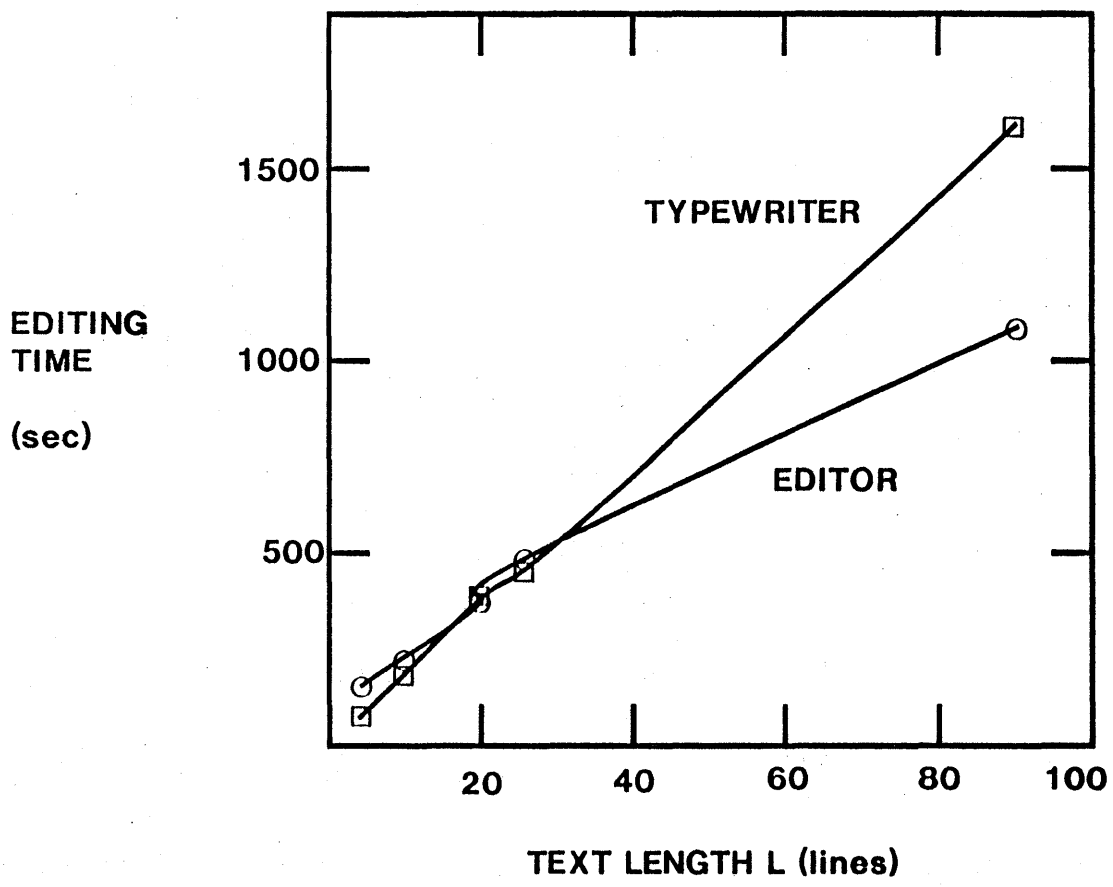


Figure 3.7. Results of experiment

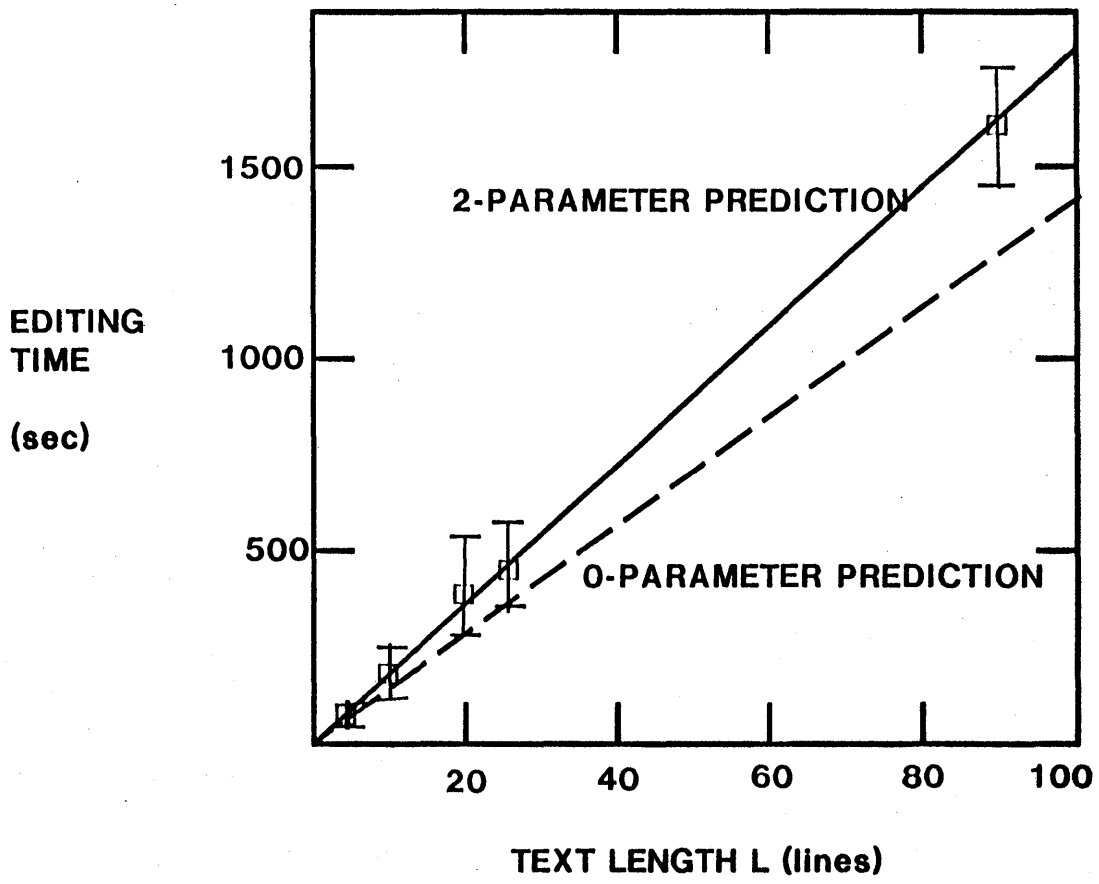


Figure 3.8. Fit of typing model to experimental data

TABLE 3.4

COMPARISON OF TYPING MODEL
$T_t = T_{st} + LT_e$ WITH DATA

	SOURCES OF ERROR				
	PARAMETERS			MODEL	TOTAL
TEXT	T_{st}	T_e	SubTotal		
T1	+27%	-18%	9%	0%	9%
T2	+11%	-19%	-8%	-2%	-10%
T3	+4%	-20%	-16%	-5%	-20%
T4	+4%	-20%	-16%	+3%	-13%
T5	ff0%	-20%	-20%	0%	-20%
Mean	9%	-19%	-10%	-1%	-11%

model. It allows us to partition the blame for errors in the model between errors in estimating the parameters and errors in the form of the equations. In order to make two-parameter predictions, estimates of the parameters were made from regressions on the RAND data. A comparison between the parameters estimated in this way and the values assumed for making the predictions is given in Table 3.3. The estimates were farthest off (649% and 86%) for the setup times T_{st} and T_{se}. They were much closer (22% and 20%) for the rate parameters T_l and T_u.

Typing Model

Figure 3.8 compares the predicted and observed times for the typing model, Equation 3.2. The zero-parameter prediction is indicated by a dotted line, while the two-parameter prediction is indicated by a solid line. When the actual typing rates of the subjects are used in the equation, the fit to the data is excellent. Using the Taylor approximation of the model, Equation 3.9, we can assign blame for the sources of error in the zero-parameter prediction. These errors are tabulated for each text in Table 3.4. On the average, the prediction was about 12% too low. Almost all of this error (10%) resulted from the error in correctly estimating the parameters; only 2% resulted from lack of fit between the

TABLE 3.5

COMPARISON OF WYLBUR MODEL
$T_e = T_{se} + \rho L T_u$ WITH DATA

	SOURCES OF ERROR				
	PARAMETERS			MODEL	TOTAL
TEXT	T_{se}	T_u	SubTotal		
T1	-67%	+4%	-63%	+46%	-47%
T2	-51%	+9%	-42%	+17%	-33%
T3	-47%	+11%	-36%	-17%	-46%
T4	-35%	+14%	-21%	-13%	-32%
T5	-13%	+22%	+9%	+1%	+10%
Ave.	-43%	+12%	-31%	+7%	-24%

model and the data. Although the estimate for the T_{st} parameter was much worse than the estimate for T_p, T_l was the source for twice as much error as was T_{st} (19% to 9%). Since the errors were in opposite directions, they partially offset each other.

Editing Model

The editing model of Equation 3.3 is compared with the observed times in Figure 3.9. Again there is a good fit between the observed and predicted editing times. Analyzing the fit in terms of the Taylor approximation (see Table 3.5), the model was about 24% too low. Again, errors in estimating the input parameters were responsible for considerably more error (31%) than was lack of fit to the model (19%). This time the major source of errors in estimating the parameters was from underestimating the setup time of the editor. It is instructive to note the frequency with which the various sources of errors partially cancel each other.

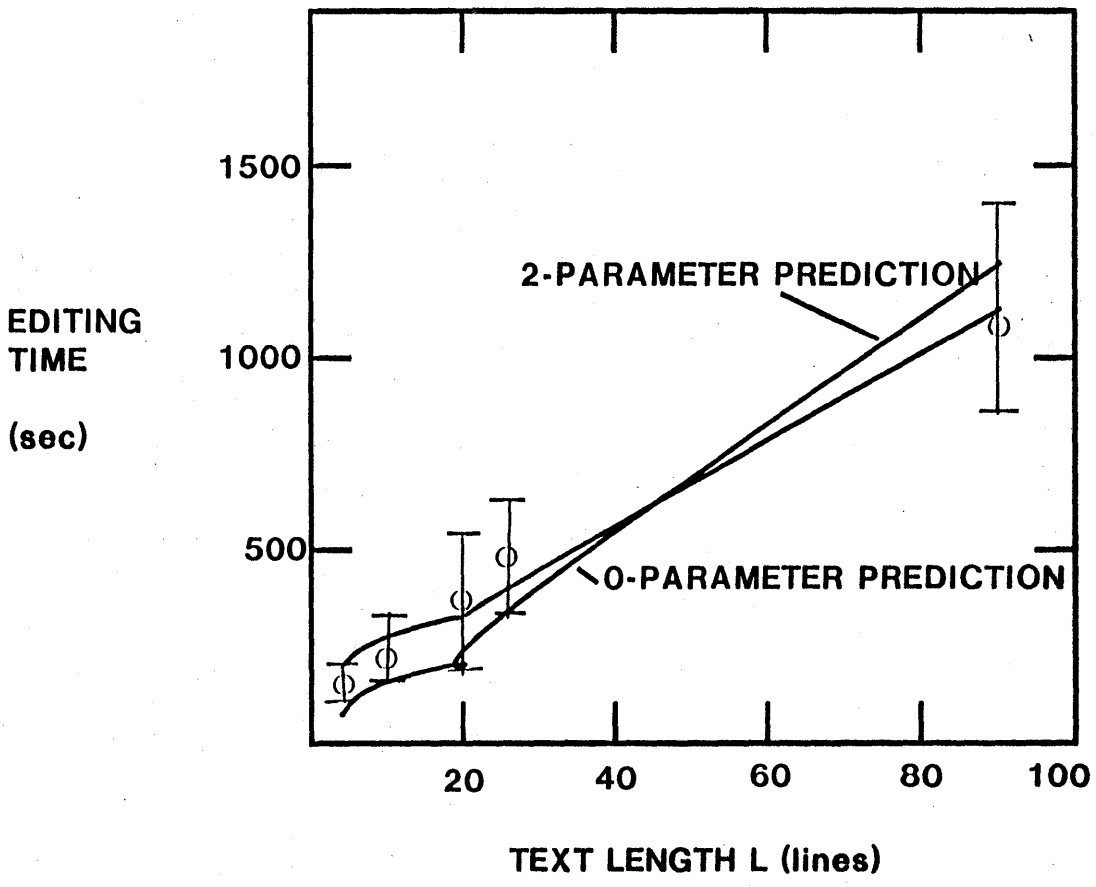


Figure 3.9. Fit of editing model to experimental data

3.5 CONCLUSIONS

The point of the forgoing exercise was to explore as a sort of case study how much insight could be gained from simple models. There were two main results.

First, with the simplest imaginable model it was possible to produce several predictions leading to practical insight. A formula for the length crossover point L_c was produced showing its functional dependence on other associated variables. A related concept of a density crossover point ρ_c was identified and expressed in functional form. It was possible to predict some unfortunate consequences of an unlucky choice in modification density for the experiment. In fact, without the insight of this derivation, the results of the experiment would have been difficult to interpret at all.

Second, the major errors in the predictions made by these simple models did not result because they were too simple, but because of errors in the values of the input parameters. For these predictions, a more sophisticated model would have been useful only to the extent it allowed one to escape the dependence on such uncertainly determined input parameters.

The sensitivity analysis identified those parts of the prediction in which little confidence could be placed. It also allowed credit and blame to be assigned after the data were in.

NOTES

[1]This number is slightly different from the numbers listed in Table 2.2, since those numbers reflect a later re-analysis of the video tapes. In order to preserve the original predictions, the originally measured parametric estimate for T_u is used in this chapter.

4

The GOMS Model of Manuscript Editing†

In the previous chapters we have analyzed tasks with a constant time/modification model. In this chapter we will break up the modification cycle into smaller pieces to refine our understanding of the process.

The questions we now address are (1) Can the behavior of a user in a manuscript editing task be described as the combination and recombination of a small number of elementary behavioral acts, much as many molecules come from few atoms? (2) If so, can we predict the stream of these acts from an analysis of the editing task environment? Can we measure the time required by the elementary acts and use them

† Adapted from Card, Moran, and Newell (1976)

to predict the time to do an editing task? Finally, (3) what grain size is appropriate for choosing the acts? Should they be at the level of a single modification in the text? or at the level of an individual motor movement?

This chapter considers a family of information-processing models. Each breaks the modification cycle at a different grain size. Their predictions are compared in detail with the behavior of a user employing the POET editor on the manuscript editing task.

4.1 THE GOMS MODEL

Previous chapters said nothing about the mechanism in the user which allow him to perform the editing task. The present chapter proposes, as a theory of the user, that his performance can be described by a set of Goals, a set of Operators, a set of Methods for achieving the goals, and a set of Selection Rules for choosing among a goal's competing methods. For short, we shall call a model specified by such components a GOMS model.

Example: Model M4B

As an example of the basic concepts of a GOMS model and the notation used, let us consider a particular model, called M4B, of the manuscript-editing task. According to the model, the user begins with the top level goal:

GOAL: EDIT MANUSCRIPT.

It is a characteristic of manuscript-editing that the larger task of editing the manuscript is composed of a collection of small edit tasks, called *unit tasks*, that are almost completely independent of each other. Thus, the obvious method for accomplishing the top level goal is to go through the individual unit tasks one by one:

GOAL: EDIT MANUSCRIPT
  GOAL: EDIT UNIT-TASK *repeat until no more unit tasks.*

In this paper, we will present the control structure of our models by displaying them in the form of an indented outline (i.e., a tree structure) of goals and operators. The outline indicates the order in which the goals are set up and the operators evoked. We will also indicate in the outline the places where methods must be selected and informally annotate it to indicate the conditionality of the various goals and operators within a

method. (A method is not always explicitly named in the outline, especially when it is the only method for accomplishing a goal; only its constituent goals and operators are shown.) The indentation above indicates that **GOAL: EDIT UNIT-TASK** is a subgoal of **GOAL: EDIT MANUSCRIPT** and the annotation in italics says that the subgoal is to be invoked repeatedly until no more unit tasks remain.

To edit a unit task, the user must first get the unit task from the manuscript and then do what is demanded by the unit task:

GOAL: EDIT UNIT-TASK
  GOAL: GET UNIT-TASK
  GOAL: DO UNIT-TASK

Each subgoal will itself evoke appropriate methods. There is a simple method for getting a task:

GOAL: GET UNIT-TASK
  GET-NEXT-PAGE *if at end of manuscript page*
  GET-UNIT-TASK

The operator **GET-NEXT-PAGE** is evoked only if there are no more edit instructions on the current page of the manuscript. The bulk of the work towards the goal—looking at the manuscript, finding editing instruction, and interpreting the instruction as an edit task—is done by the operator **GET-UNIT-TASK.**

To do a unit task in POET there is a two-step method:

GOAL: DO UNIT-TASK
  GOAL: LOCATE LINE
  GOAL: MODIFY TEXT

The POET editor must first be located at the line where the correction is to be made, and then the appropriate text on that line is modified.

To locate POET at a line, there is a choice of two methods:

GOAL: LOCATE LINE
  [select: USE-LF-METHOD
  USE-QS-METHOD]

To use the **LF-METHOD**, the linefeed key is pressed repeatedly, causing the editor to advance one line forward each time. To use the **QS-METHOD**, a string in quotation marks is typed which identifies the line. Usually the **LF-METHOD** is selected when the new unit task is within a few lines of the previous unit task, and the **QS-METHOD** is selected when the new unit task is farther away.

Once the line has been located, there is a choice of how to modify the text:

```
GOAL: MODIFY TEXT
   [select:  USE-S-CMD
             USE-M-CMD]
   VERIFY-EDIT
```

That is, either POET's Substitute command or Modify command can be used to alter text on a line, but in either case a **VERIFY-EDIT** operation is evoked to check what actually happened against the user's intentions.

Putting all the pieces together into one tree structure, we have:

```
GOAL: EDIT MANUSCRIPT
. GOAL: EDIT UNIT-TASK repeat until no more unit tasks
. . GOAL: GET UNIT-TASK
. . . GET-NEXT-PAGE if at end of manuscript page
. . . GET-UNIT-TASK
. . GOAL: DO UNIT-TASK
. . . GOAL: LOCATE LINE
. . . . [select:  USE-QS-METHOD
                   USE-LF-METHOD]
. . . GOAL: MODIFY TEXT)
. . . . [select:  USE-S-CMD
                   USE-M-CMD]
. . . . VERIFY-EDIT
```

The dots at the left of each line show the depth of the goal stack.

To complete this model of manuscript-editing, we must add method selection rules that would determine the actual sub-methods at the two occurrences of "select". Due to the simplicity of the structure of the methods, we have used a simplified programming notation with the conditions written off to the right as notes. These conditions are not merely comments on the model, but are an integral part of the model, as are the selection rules.

The step by step behavior of the model in performing a unit task is traced in Table 4.1. The user is imagined to have a goal stack with the current goal being on top of the stack. New subgoals are pushed onto the stack and completed goals (whether satisfied or abandoned) are popped off the stack. The goals eventually cause operators to be executed. It is during the execution of the operators that interactions with the physical world task place. The user executes the operator **GET-UNIT-TASK** by turning to the manuscript, scanning it until he finds the

TABLE 4.1

TRACE OF MODEL M4B DURING PERFORMANCE OF A UNIT TASK

Step	Contents of Goals Stack[a]	Operator Executed	User Action
1	(EDIT MS)[b]		
2	(EDIT MS) (EDIT UT)		
3	(EDIT MS) (GET UT) (EDIT UT)		
4	(EDIT MS) (GET UT) (EDIT UT)	GET-UT	*(Looks at manuscript)*
5	(EDIT MS) (GET UT)		
6	(EDIT MS) (GET UT) (DO UT)		
7	(EDIT MS) (GET UT) (DO UT) (LOCATE LINE)		
8	(EDIT MS) (GET UT) (DO UT) (LOCATE LINE)	USE-LF-METHOD	*(Types* <LF>*)*
9	(EDIT MS) (GET UT) (DO UT)		
10	(EDIT MS) (GET UT) (DO UT) (MODIFY-TEXT)		
11	(EDIT MS) (GET UT) (DO UT) (MODIFY-TEXT)	USE-S-CMD	*(Types* sIdi<CR>idi<CR><CR>*)*
12	(EDIT MS) (GET UT) (DO UT) (MODIFY-TEXT)	VERIFY-EDIT	*(Types* /*)*
13	(EDIT MS) (GET UT) (DO UT)		
14	(EDIT MS) (GET UT)		
15	(EDIT MS)		

[a]Top of stack, that is, the current goal, is at the right

[b]To save space, the word **GOAL:** has been dropped from the beginning of goal expressions, **MANUSCRIPT** has been abbreviated **MS**, and **UNIT-TASK** has been abbreviated **UT**.

next task, reading the instructions, and turning back to the terminal. The user executes the operator **USE-S-CMD** for the second task in Plate 1.2 by typing `sIdi<CR>idi<CR><CR>` as described in the previous section.

Components of the GOMS Model

Goals. A goal is a symbolic structure that defines a state of affairs to be achieved and determines a set of possible methods by which it may be accomplished. In the example, the goals are **GOAL: EDIT MANUSCRIPT**, **GOAL: EDIT UNIT-TASK**, **GOAL: GET UNIT-TASK**, **GOAL: DO UNIT-TASK**, **GOAL: LOCATE LINE**, and **GOAL: MODIFY TEXT**. The dynamic function of a goal is to provide a memory point to which the system can return on failure or error and from which information can be obtained about what is desired, what methods are available, and what has been tried already.

Operators. Operators are elementary motor or information-processing acts, whose execution is necessary to change any aspect of the user's memory or to affect the task environment. In the example, the operators are: **GET-NEXT-PAGE**, **GET-UNIT-TASK**, **USE-QS-METHOD**, **USE-LF-METHOD**, **USE-S-CMD**, **USE-M-CMD**, and **VERIFY-EDIT**. The behavior of the user is ultimately recordable as a sequence of these operations. In the example traced in Table 4.1, the sequence of behavior is **GET-UNIT-TASK**, **USE-LF-METHOD**, **USE-S-CMD**, **VERIFY-EDIT**. The model does not deal with any fine structure of concurrent operation.

An operator is defined by a specific effect (output) and by a specific duration. The operator may take inputs, and its outputs and duration may be a function of its inputs. An obvious example is a typing operator, whose input is the text to be typed, whose output is the keystroke sequence to the keyboard, and whose duration is (approximately) a linear function of the number of characters.

For a specific model the operators define the grain of analysis. In general, they embody an indeterminate mixture of basic psychological mechanisms and learned organized behavior, the mixture depending on the level at which the model is cast. The finer the grain of analysis, the more the operators reflect basic psychological mechanisms. The coarser the grain of analysis, the more the operators reflect the specifics of the task environment, such as the terminal, the physical arrangement, and the editor.

Methods. A method describes a procedure for accomplishing a goal. The description of the procedure is cast as a conditional sequence of goals and operators, with conditional tests on the contents of the user's immediate memory and on the state of the task environment. In the

example above, one of the methods was

GOAL: GET UNIT-TASK
GET-NEXT-PAGE *if at end of manuscript page*
GET-UNIT-TASK

This method is associated with the goal **GOAL: GET UNIT-TASK.** It will give rise to either the operator sequence **GET-NEXT-PAGE, GET-UNIT-TASK** or the operator sequence **GET-UNIT-TASK** depending on whether the test *at end of manuscript page* is true of the task environment at the time the test is performed.

In the manuscript-editing task, the methods are sure of success, up to the possibility of having been mis-selected, the occurrence of errors of implementation, and the reliability of the equipment. By contrast, in problem solving tasks, such as the task faced by a novice in the Tower of Hanoi puzzle, methods have a chance of success distinctly less than certainty, due to the user's lack of knowledge or appreciation of the task environment. This uncertainty is a prime contributor to the problem solving character of a task; its absence is a characteristic of a routine cognitive skill.

Methods are learned procedures which the user has at performance time. They are not plans that are created during a task performance. They constitute one of the major ways in which familiarity (skill) expresses itself. The particular methods that the user builds up from prior experience, analysis, and instruction reflect the detailed structure of the task environment. In the manuscript-editing task, they reflect knowledge of the exact sequence of steps required by the editor to accomplish specific tasks.

Control Structure: Selection Rules. When a goal is attempted, there may be more than one method available to the user to accomplish the goal. The selection of which method is to be used need not be an extended decision process, for it may be that the task environment features dictate that only one method is appropriate. On the other hand, a genuine decision may be required. The essence of skilled behavior is that these selections are not problematical, that they proceed smoothly and quickly and without the eruption of puzzlement and search characteristic of problem solving behavior.

In the GOMS model, method selection is handled by a set of *selection rules.* Each selection rule is of the form "if such and such is true in the current task situation, then use method M". Selection rules for the **LOCATE** goal of the example model might read: *if the number of lines to the next modification is less than 3, use the* **LF-METHOD;** *else use the* **QS-METHOD.** Such rules allow us to predict from knowledge of the task environment, in this case the number of lines to

the target, which of several possible methods will be selected by the user in a particular instance.

Limitations of the GOMS Model

For error-free behavior, the GOMS model provides a complete dynamic description of behavior, measured at the level of goals, methods, and operators. Given a specific task (i.e., a specific manuscript), this description can be expanded into a sequence of operations (operator executions). By associating times with each operator, such a model will make total time predictions. If these times are given as distributions, it will make statistical predictions. But, without augmentation, the model will not make predictions if errors occur. Yet, errors exist in routine cognitive skilled behavior. Indeed, error rates may not even be small, in the sense of having negligible frequency, taking negligible time, or having negligible consequences. What is true of skilled behavior is that the detection and correction of errors is mostly routine. It cannot be entirely routine, since rare types of errors for which the user is unprepared are always possible (e.g., the terminal catching fire, the editor performing incorrectly, etc.). But, in the main, errors are quickly detected and converted to the additional time to correct the error. The final result of the behavior remains relatively error free and can be characterized solely by the time to completion. Thus, errors can be converted to variance in operator times, so that the GOMS theory can be applied to actual behavior at the price of degraded accuracy.

To cover the full range of human behavior, the general theory of the human information processing systems requires a more flexible control structure, such as a *production system* (Newell & Simon, 1972; Newell, 1973). In fact, some of the models to be described have been expressed as production systems and executed as computer simulations. Since in this chapter the analysis will not be carried deeply into the behavior of errors, the GOMS model contains an adequate specification of control.

Model Grain

The example model displayed above is not the only possible GOMS model for the manuscript editing task. Models could be given with either more or less detail. Thus, there is an important issue of the appropriate *grain* of the analysis.

A priori, it is not possible to know which grain size is appropriate. As the grain of the analysis becomes finer, the model successively accumulates opportunities for conditional behavior (either optional application of some method or differentiation into cases). Thus, from one point of view, models at finer grain should be more accurate. But at

a finer grain the low-level operators are combined to form functional units that a coarser grain analysis would reflect directly. It has been known for some time (Abruzzi, 1956; Smith & Harris, 1954), that the time required for an operator in a sequence of operators, especially one in a fine grain analysis, *may* depend on other operators in the sequence. Furthermore, there is typically greater error in the measurement of finer grain operators than of coarser grain operators. Thus, a finer grain analysis might actually be less accurate.

In order to see how the grain of analysis affects the accuracy of the models, we will redo the analysis at several levels of detail, comparing the resulting models. There appear to be two essentially independent dimensions along which the grain of analysis can be made finer or coarser. The primary dimension involves the duration of the operators. Given that human primitive operators are something less than a tenth of a second, many levels of time aggregation are possible. The second dimension involves the amount of differentiation between operators, i.e., the degree to which conditionality is suppressed and alternative operators (or sequences of operators) are considered to be the same operator. Such case-analysis aggregation can happen at any level of time aggregation.

We will explore variations of GOMS models along both of these dimensions. Table 4.2 describes the family of nine manuscript-editing models we will consider, and Tables 4.3 to 4.5 lay out the models themselves. Each model is given a name of the form M*ld*. The *l* indicates the *level*, i.e., the order of magnitude time grain of the model in seconds. For convenience, we consider models which increase roughly in powers of 2 sec. Thus, M16A has operators whose durations are on the order of 2^4 seconds; M0.5A has operators closest to 2^{-1} sec. This latter is the measurement limit in the experimental arrangement.

Within each level we consider various degrees of differentiation. There is no convenient metric here, nor do differentiations at the different levels correspond, so we simply assign arbitrary letter labels—the *d* in the model name. However, we do adopt the convention that model M*l*A is the most aggregated model at level *l*, i.e., the one that collapses conditional sequences into each other as much as possible.

At the most aggregated level, Model M16A, shown in Table 4.3, consists of a single operator, **EDIT-UNIT-TASK**. The goal of manuscript-editing is accomplished by repeating this operator for each unit task. With a single operator, M16A always predicts that it takes the same amount of time to do a unit task, hence the same amount of time to do a total job of *n* unit tasks. Level 4 models come from decomposing the unit task into its invariant functional cycle: (1) get the next edit task, (2) locate the editor at the line on which the correction is

TABLE 4.2

DESCRIPTION OF GOMS MODELS TESTED

Level 16 Models	
M16A	Constant time per unit task. Only one operator: **EDIT-UNIT-TASK.**
Level 4 Models	
M4A	Single operator for each functional step in unit task sequence: **GET-UNIT-TASK, LOCATE-LINE, MODIFY-TEXT, VERIFY-EDIT.**
M4B	Like M4A but with operators **LOCATE-LINE** and **MODIFY-TEXT** broken into separate cases based on methods used to accomplish them.
Level 2 Models	
M2A	Like M4B, but with operators at the level of typing a system command **(SPECIFY-CMD)** or typing an argument to a command **(SPECIFY-ARG).**
M2B	Like M2A but with **SPECIFY-CMD** and **SPECIFY-ARG** broken into separate cases according to whether they involve an implicit need for use to get information from manuscript (suffix **/G**) or not (suffix **/NG**).
M2C	Like M2A but with **SPECIFY-CMD** and **SPECIFY-ARG** broken into separate cases according to four method contexts: quoted string method (**/Q**), first argument to substitute command (**/S1**), second argument to substitute command (**/S2**), or modify command (**/M**).
M2D	Like M2A, but with all the distinctions in both M2B and M2C combined multiplicatively.
Level 0.5 Models	
M0.5A	Like M2B, but with operators at the level of basic perceptual, cognitive, and motor actions: **LOOK-AT, HOME, TURN-PAGE, TYPE,** and **MOVE-HAND.** All mental actions not overlapped with motor operations represented as **MENTAL** operator.
M0.5E	Like M0.5A, but with **MENTAL** broken down into **SEARCH-FOR, COMPARE, CHOOSE-CMD,** and **CHOOSE-ARG.**
M0.5E'	Like M0.5E, but typing time is a constant (i.e., not parameterized by the number of keystrokes).

TABLE 4.3

LEVEL 16 AND LEVEL 4 MODELS

Model M16A:

```
(GOAL: EDIT MANUSCRIPT)
.   (EDIT-UNIT-TASK)                    . repeat until no more unit tasks
```

Model M4A:

```
(GOAL: EDIT MANUSCRIPT)
.   (GOAL: EDIT UNIT-TASK)              . repeat until no more unit tasks
.   .   (GOAL: GET UNIT-TASK)           . . if task not remembered
.   .   .   (GET-NEXT-PAGE)             . . . if at end of manuscript page
.   .   .   (GET-UNIT-TASK)
.   .   (GOAL: DO UNIT-TASK)            . . if an edit task was found
.   .   .   (LOCATE-LINE)               . . . if task not on current line
.   .   .   (MODIFY-TEXT)
.   .   .   (VERIFY-EDIT)
```

Model M4B:

```
(GOAL: EDIT MANUSCRIPT)
.   (GOAL: EDIT UNIT-TASK)              . repeat until no more unit tasks
.   .   (GOAL: GET UNIT-TASK)           . . if task not remembered
.   .   .   (GET-NEXT-PAGE)             . . . if at end of manuscript page
.   .   .   (GET-UNIT-TASK)
.   .   (GOAL: DO UNIT-TASK)            . . if an edit task was found
.   .   .   (GOAL: LOCATE LINE)         . . . if task not on current line
.   .   .   .   [select (USE-QS-METHOD)
.   .   .   .           (USE-LF-METHOD)]
.   .   .   (GOAL: MODIFY TEXT)
.   .   .   .   [select (USE-S-CMD)
.   .   .   .           (USE-M-CMD)]
.   .   .   .   (VERIFY-EDIT)
```

to be made, (3) modify the line of text, and (4) verify that the edit was done correctly. Level 2 models arise by decomposing the methods used at Level 4 into the individual steps of specifying commands and arguments (see Table 4.4).

Both Levels 4 and 2 are driven by the structure of the POET commands. These are themselves reflections of the demands of the task as it is defined in the manuscript. At Level 0.5 an entirely different set of operators (see Table 4.5) comes into view which are not defined functionally by their role in a command language, but are defined by reference to the basic physical and mental actions of the user—typing, looking, moving a hand, and various mental operations. These operators, unlike the operators at other levels, are task free.

The cost of obtaining the estimates of all the different operators and selection rules increases as the size of the operators decrease, because more data is required for a given level of robustness and because the observation and measurement problems increase at the lower levels. A possible compensation for the greater cost of using the Level 0.5 operators is that, unlike the larger operators, it may not be necessary to determine lower level operators for each experimental task.

4.2 EXPERIMENTAL METHOD

In order to evaluate the usefulness of the GOMS model for describing user behavior in the manuscript editing task, an experiment was conducted in which detailed observations were made of users editing a prescribed manuscript. Although this is a laboratory setting, an effort was made to make the situation was naturalistic from the users' point of view: the physical surroundings, the task, the terminal, and the editor were familiar as part of the users' daily activities. The manuscript and the modifications to be made on it were selected to be typical.

Subjects. Subjects were four employees of the Xerox Palo Alto Research Center. Three (S4, S13, and S22) were or had been professional secretaries, one (S1) was a programmer. All had at least a year of daily experience using the editor.

Manuscript. The manuscript was an 11 page memo. Each page was 8-1/2 by 11 inches, with 55 lines of text and 70 characters per line, printed unjustified in a 10-point fixed-pitch font. There were 73 different modifications marked with a red pen, giving an average density of one modification every 8.3 lines, or 6.6 modifications per page (from 3 to 11 on any one page). An effort was made to vary the number of lines

TABLE 4.4

LEVEL 2 MODELS

Model M2A:

```
(GOAL: EDIT MS)
.  (GOAL: EDIT UT)                                  . repeat until no more unit tasks
.  .  (GOAL: GET UT)                                . . if task not remembered
.  .  .  (GET-NEXT-PAGE)                           . . . if at end of manuscript page
.  .  .  (GET-FROM-MS)
.  .  (GOAL: DO UT)                                 . if an edit task was found
.  .  .  (GOAL: LOCATE LINE)                       . . . if task not on current line
.  .  .  .  [select  (GOAL: USE QS-METHOD)
.  .  .  .           .  (SPECIFY-CMD)
.  .  .  .           .  (SPECIFY-ARG)
.  .  .  .           (GOAL: USE LF-METHOD)
.  .  .  .           .  (SPECIFY-CMD)]             . . . . . repeat until at line
.  .  .  .  (VERIFY-LOC)
.  .  .  (GOAL: MODIFY TEXT)
.  .  .  .  [select  (GOAL: USE S-CMD)
.  .  .  .           .  (SPECIFY-CMD)
.  .  .  .           .  (SPECIFY-ARG)
.  .  .  .           .  (SPECIFY-ARG)
.  .  .  .           (GOAL: USE M-CMD)
.  .  .  .           .  (SPECIFY-CMD)
.  .  .  .           .  (SPECIFY-CMD)              . . . . . repeat until at text
.  .  .  .           .  (SPECIFY-ARG)
.  .  .  .           .  (SPECIFY-CMD)]
.  .  .  .  (VERIFY-EDIT)
```

Model M2B:

```
SPECIFY-CMD     = = >   SPECIFY-CMD/G, SPECIFY-CMD/NG

SPECIFY-ARG     = = >   SPECIFY-ARG/G, SPECIFY-ARG/NG
```

Model M2C:

```
SPECIFY-ARG     = = >   SPECIFY-ARG/Q, SPECIFY-ARG/M,
                = = >   SPECIFY-ARG/S1, SPECIFY-ARG/S2
```

Model M2D:

```
SPECIFY-CMD     = = > SPECIFY-CMD/G, SPECIFY-CMD/NG

SPECIFY-ARG     = = > SPECIFY-ARG/Q/G, SPECIFY-ARG/Q/NG,
                = = > SPECIFY-ARG/M/G, SPECIFY-ARG/M/NG,
                = = > SPECIFY-ARG/S1/G, SPECIFY-ARG/S1/NG,
                = = > SPECIFY-ARG/S2/G, SPECIFY-ARG/S2/NG
```

TABLE 4.5

MODEL M0.5E

```
(GOAL: EDIT MS)
.   (GOAL: EDIT UT)                                  . repeat until no more unit tasks
.   .   (GOAL: GET UT)                               . . if task not remembered
.   .   .   (GOAL: TURN PAGE)*                      . . . if at end of manuscript page
.   .   .   (GOAL: GET-FROM MS)*
.   .   (GOAL: DO UT)                                . . if an edit task was found
.   .   .   (GOAL: LOCATE LINE)                     . . . if task not on current line
.   .   .   .   (CHOOSE-CMD)
.   .   .   .   [select  (GOAL: USE QS-METHOD)
.   .   .   .           .   (GOAL: SPECIFY-CMD)*
.   .   .   .           .   (GOAL: SPECIFY ARG)*
.   .   .   .           (GOAL: USE LF-METHOD)
.   .   .   .           .   (GOAL: SPECIFY-CMD)*    . . . . . repeat until at line
.   .   .   .   (GOAL: VERIFY LOCATION)*
.   .   .   (GOAL: MODIFY TEXT)
.   .   .   .   (CHOOSE-CMD)
.   .   .   .   [select  (GOAL: USE S-CMD)
.   .   .   .           .   (GOAL: SPECIFY CMD)*
.   .   .   .           .   (GOAL: SPECIFY ARG)*
.   .   .   .           .   (GOAL: SPECIFY ARG)*
.   .   .   .           (GOAL: USE M-CMD)
.   .   .   .           .   (GOAL: SPECIFY CMD)*
.   .   .   .           .   (GOAL: SPECIFY CMD)*    . . . . . repeat until at text
.   .   .   .           .   (GOAL: SPECIFY ARG)*
.   .   .   .           .   (GOAL: SPECIFY CMD)*]
.   .   .   .   (GOAL: VERIFY EDIT)*
*(GOAL: TURN PAGE)
.   (LOOK-AT MS)                                     . repeat twice
.   (ACTIVE)
.   (MOVE-HAND)                                      . repeat twice
.   (TURN-PAGE)
*(GOAL: GET-FROM MS)
.   (LOOK-AT MS)
.   (SEARCH-FOR)
.   (LOOK-AT D)                                      . optional
*(GOAL: SPECIFY CMD)
.   (GOAL: GET-FROM MS)*                             . if not already selected
.   (CHOOSE-CMD)                                     . if not already selected
.   (GOAL: TYPE STRING)*
*(GOAL: SPECIFY ARG)
.   (GOAL: GET-FROM MS)*                             . optional
.   (CHOOSE-ARG)
.   (GOAL: TYPE STRING)*
*(GOAL: VERIFY)
.   (LOOK-AT D)
.   (GOAL: GET-FROM MS)*                             . optional
.   (COMPARE)
*(GOAL: TYPE STRING)
.   (HOME)                                           . optional
.   (LOOK-AT K)                                      . optional
.   (LOOK-AT D)                                      . optional
.   (TYPE STRING)
```

between consecutive modifications and to place an equal number of modifications in each of the left, right, and middle portions of the page. The marked modifications were relatively short: four of them were deletions (of an average of 5.5 characters), 26 were insertions (of an average of 2.9 characters), and 40 were replacements (of an average of 4.1 characters by 4.4 characters). The paragraph of Figure 1.2 was taken from the manuscript and illustrates the style in which modifications were indicated to the user.

Terminal. Two terminals were used in the experiment: a Texas Instruments (TI) "Silent 700" (prints on paper at 30 characters/sec) and a CRT display 8-1/2 inches wide by 10-3/4 inches high (42 lines, with 72 characters per line). Text was displayed on the CRT at a maximum rate of 6 lines per sec. The display was programmed to operate according to a simple scrolling discipline (the same discipline used on the hardcopy terminal): each new line was displayed at the bottom of the screen with the other lines scrolling up to make room, i.e., the last 42 lines of an interaction were visible on the screen.

Measurement apparatus. The terminal was connected to a large computer running the POET editor under the TENEX time-sharing system. For this experiment the terminal was modified to time- stamp and record on a data file all input and output events. It should be noted that the accuracy of the timing of events did not depend on the response of the time-sharing system. Accuracy of time-stamping was to within 32 msec of the actual time of the event at the terminal. The average response time of the editor to commands during the experiment was 0.8 sec (SD = 0.6 sec).

Two television cameras were focussed on the user, one camera giving an overall view of the situation, the other closely focussed on the user's face from which it could be determined whether he was looking at the manuscript, the keyboard, or the CRT. Pictures from the cameras were electronically combined to form a single split image recorded on videotape. The user wore a lapel microphone, recording onto the soundtrack of the videotape. A digital clock was electronically mixed with the video picture, time-stamping each frame. The times measured from video frames were accurate to 33 msec (one video frame).

Procedure. The user was seated before the terminal with the manuscript to his left. He first performed a short editing task on another manuscript for warmup and to insure that he understood what to do. In the first two sessions run, the users were instructed to proceed through the manuscript inserting an asterisk at the beginning of the line, since these two were run to investigate only methods for locating the target line. For the other three sessions, the users were instructed to edit the 11 page manuscript, which took approximately 20 minutes.

4.3 RESULTS

4.31 Operator Sequences

Is it possible to predict the actual sequence of operators a person will use to do the task? In order to predict operator sequences, it is necessary to be able to predict method selections. Consequently, the method selections of the users were investigated. To get an indication how well the models, in concert with the selection rules, could predict operator sequences, one user's protocol was singled out for intensive analysis. Predictions were made of the operator sequence she would use for each of the tasks in the manuscript. The sequence of operators the user actually employed was determined and the two compared. The comparison was repeated for each model.

Method Selection Rules

As we saw in the description of model M4B and as indicated by the "select" in Table 4.3, there are two places where, for a given goal, the user has a choice of methods. The first method selection comes in deciding how to "locate the line," that is, how to make the Current Line of the editing system be the line containing the text to be modified (the **LOCATE** goal). The second method selection comes in choosing between commands for making the text modification (the **MODIFY** goal). The users' behavior in the five experimental sessions was examined to identify the methods they employed in the service of these goals. Table 4.6 gives the methods observed and the frequencies with which the methods were selected. **QS-METHOD** and **LF-METHOD** are the methods previously described for the **LOCATE** goal. **S-CMD** and **M-CMD** are the methods previously described for the **MODIFY** goal The others are additional methods that were used less frequently and may be described as follows:

> **+N-METHOD.** The user estimates the number of lines, *n*, to the next unit task then types the command *+n/*, which causes POET to advance *n* lines and print the line. It is assumed that a correction may be needed, e.g., the user may have to type a few linefeeds (each of which move's him down a line), ↑'s (each of which moves him up a line), or may even have to repeat the *+n/* command with a new *n*.

TABLE 4.6

FREQUENCY OF METHOD SELECTIONS

	Experimental Session				
	1	2	3	4	5
User	S1[a]	S4	S4	S22	S13
Terminal	TI	TI	CRT	CRT	CRT
***LOCATE** Methods:*					
QS-METHOD	44 (65%)	1 (2%)	0 (0%)	40 (62%)	46 (68%)
LF-METHOD	11 (16%)	14 (21%)	45 (68%)	25 (38%)	21 (31%)
+N-METHOD	2 (3%)	51 (77%)	20 (30%)	0 (0%)	0 (0%)
AN-METHOD	11 (16%)	0 (0%)	1 (2%)	0 (0%)	1 (1%)
***MODIFY** Methods:*					
S-CMD	[b]	48 (73%)	[b]	57 (86%)	63 (93%)
M-CMD	[b]	18 (27%)	[b]	9 (14%)	4 (6%)
C-CMD	[b]	0 (0%)	[b]	0 (0%)	1 (1%)

[a]S1 was the only subject who was a programmer.
[b]No **MODIFY** method data was collected.

AN-METHOD. The user selects an easily specified "anchor" line near the target line, e.g., a blank line (specified by the empty string ""), the last line of a page (which has the special symbol $), or a line that has a short unique string, such as a paragraph number. Then the target line is reached by using linefeeds or ↑'s. For example, the command ""linefeed locates POET at the first line of the next paragraph.

A striking feature of the numbers in Table 4.6 is each user clearly has a default method. By knowing only the default method of the user, his method selection can be predicted correctly about 68% of the time for the **LOCATE** goal and 84% of the time for the **MODIFY** goal. Apparently, the user will use this default method unless it is obviously inefficient (as in linefeeding a line at a time through ten pages of text to get the next task).

By taking into account other features of the task environment, the prediction of which method the user will select can be improved. The most important characteristic of the task environment to consider for **GOAL: LOCATE** methods is the number of lines D between the Current Line and the line with the text to be modified next. As is clear from Table 4.7, all users used the **LF-METHOD** if the next line was close enough. Where the users differed was in the threshold for how far away the target had to be before they shifted to other methods. The time required to use the **LF-METHOD** is sensitive to the speed of the terminal. (Each time the user types linefeed, the system prints out the new Current Line). It is not surprising, therefore, that the threshhold for when to abandon the **LF-METHOD** is lower when the user is using a slow terminal than when he is using a fast one. For the slow 30 char/sec TI terminal, both users shifted at $D = 3$ lines. For the faster display terminals, two users shifted at $D = 5$ lines. The other user (for whom the **LF-METHOD** was the default) held out until $D = 10$ lines.

The complete prediction of which method each user will employ for the **LOCATE** goal is organized as a set of Selection Rules in Table 4.8. Each row gives the results of the accumulation of rules R1 to Rn adding rules one at a time. The Hits column shows the total number of cases correctly predicted, Misses shows the number of cases in which the prediction was wrong (Hits + Misses = the total number of method selections). As each rule is added, the set of rules taken together predicts more cases correctly, but a few individual cases which were predicted correctly may now be missed. For example, adding rule R2 in the second line of the table correctly predicts 11 method selections of the 24 that had been missed using Rule R1 alone, but at the cost of missing 2 of the 44 that were previously hits—a net gain of 9. As the table shows,

TABLE 4.7

FREQUENCY OF LOCATE METHODS BY dY

User	Method	Number of lines from current line to line containing target (*D*)										
		1	2	3	4	5	6	7	8	9	10-14	15+
S1 (TI)	LF	8	3	[a]								
	QS		2	4	5	2		1	3	4	8	15
	+N		1	1								
	AN			1	2				1	1	3	3
S4 (TI)	LF	8	4	1		1						
	QS											1
	+N		1	5	5	3		1	4	4	11	17
	AN											
S4 (CRT)	LF	6	7	6	5	3		1	3	2	2	10
	QS											
	+N					1			1	2	9	7
	AN											1
S22 (CRT)	LF	6	5	6	5	1		1		1		
	QS		1		1	2			4	4	10	18
	+N											
	AN											
S13 (CRT)	LF	8	4	4	3	1						1
	QS			3	2	3		1	3	4	12	17
	+N											
	AN											1
MS Total[b]		8	6	6	5	4	0	1	4	4	11	19

[a]The vertical bar indicates where **LF-METHOD** stops being the preferred method

[b]Frequency of *D*'s taking the tasks over the whole manuscript in order. Since users usually did some edits in a different order, totals for different experiments in the same column are not necessarily equal.

TABLE 4.8

SELECTION RULES FOR **LOCATE** GOAL

User		Rule	This Rule Gain	This Rule Lose	Cumulative Hits	Cumulative Misses	Cumulative %Hits
S1 (TI)	R1:	Use the **QS-METHOD** unless another rule applies.	44	0	44	24	65%
	R2:	If $D < 3$, use the **LF-METHOD**	11	2	53	15	78%
	R3:	If the target line is the last line of the page, use the **AN-METHOD** (with $).	5	0	58	10	85%
	R4:	If the current method is to use paragraph numbers for search strings and the target line is near a paragraph number, then use the **AN-METHOD.**	2	0	60	8	88%
S4 (TI)	R1:	Use the **+N-METHOD** unless another rule applies.	51	0	51	15	77%
	R2:	If $D < 3$, use the **LF-METHOD.**	12	1	62	4	94%
S4 (CRT)	R1:	Use the **LF-METHOD** unless another rule applies	45	0	45	21	68%
	R2:	If $D > 9$, use the **+N-METHOD.**	16	12	49	17	74%
	R3:	If the target line is on the next page of the manuscript, use the **LF-METHOD**	56	10	56	10	85%
S22 (CRT)	R1:	Use the **QS-METHOD** unless another rule applies.	40	0	40	26	61%
	R2:	If $D < 5$, use the **LF-METHOD.**	22	2	60	6	91%
S13 (CRT)	R1:	Use the **QS-METHOD** unless another rule applies.	46	0	46	22	68%
	R2:	If $D < 5$, use the **LF-METHOD.**	19	5	60	8	88%
	R3:	If $D = 3$ or 4 and *Column* > 25, then use the **QS-METHOD.**	4	2	62	6	91%

it is possible to predict the method selection for users an average of 90% of the time using from two to four rules.

Data for Detailed Studies of User S13

In order to compare the operator sequence predicted by the models with a sequence actually employed, the recorded behavior of one user S13 was subjected to intensive analysis. S13 was a highly skilled female secretary (typing rate 103 words per minute) with about two years experience on the POET editor, much of it with the type of terminal used in this experiment.

The video-taped record of her behavior and the time-stamped file of keystrokes were combined into a protocol, a fragment of which is reproduced in Table 4.9. The protocol was coded directly from the video-tape and the keystroke file using a set of descriptive operators not related a priori to any model. The overwhelming bulk of behavior was coded by the operators **TYPE**, **LOOK-AT**, and **MENTAL** defined as follows:

TYPE(*char1 char2* ...) A burst of typewriting starting with the beginning of the finger trajectory toward the first key and ending when the last key makes contact. A "burst" is defined as a sequence of keystrokes with no more than 300 msec between successive key contacts.

LOOK-AT(*place*) Act of looking from one place to another. A *place* is either the CRT, the keyboard, or the manuscript. **LOOK-AT** includes the physical head movement and gross eye movement, but does not include any perceptual scanning within a place (such as searching a manuscript page for a new task).

MENTAL Generic operator for mental activity that does not overlap with physical operations.

Other operators, used infrequently, were **HOME(*hand place*)** for moving a hand to the keyboard preparatory to typing, **MOVE-HAND(*hand place*)** for other hand movements, **TURN-PAGE**, **ACTION(*description*)**, and **EXPRESSION(*description*)**. The last two were miscellaneous categories for recording other behavior.

The first three unit tasks were discarded before analysis to minimize any warmup effect. The remaining 70 unit tasks were partitioned into two comparable data sets: a *Derivation* data set, consisting of the 34 unit tasks on the odd-numbered pages, and a *Crossvalidation* data set con-

TABLE 4.9

PROTOCOL RECORD OF ONE UNIT TASK

Start (*min:sec*)	Stop (*min:sec*)	ΔT (*sec*)	Operator Description	
18:56.33	18:56.73	0.40	LOOK-AT(MANUSCRIPT)	
18:56.73	18:58.89	2.16	MENTAL	
18:58.89	18:59.41	0.52	HOME(LEFT)	
18:59.41	18:59.66	0.25	MENTAL	*[This protocol describes*
18:59.66	18:59.94	0.23[a]	LOOK-AT(KEYBOARD)	*the behavior during*
18:59.89	19:00.14	0.25	TYPE(")	*the last unit task*
19:00.14	19:00.24	0.10	MENTAL	*shown in Figure 1]*
19:00.24	19:00.48	0.24	LOOK-AT(CRT)	
19:00.48	19:01.11	0.63	MENTAL	
19:01.11	19:01.43	0.32	LOOK-AT(KEYBOARD)	
19:01.43	19:01.70	0.27	MENTAL	
19:01.70	19:01.82	0.12	TYPE(e)	
19:01.82	19:01.92	0.10	MENTAL	
19:01.92	19:02.66	0.07[a]	TYPE(x i s < CR > /)	
19:01.99	19:02.34	0.35	LOOK-AT(CRT)	
19:02.34	19:04.16	1.82	MENTAL	
19:04.16	19:04.53	0.37	LOOK-AT(MANUSCRIPT)	
19:04.53	19:05.48	0.95	MENTAL	
19:05.48	19:05.83	0.15[a]	LOOK-AT(CRT)	
19:05.63	19:05.91	0.28	TYPE(. s)	
19:06.06	19:06.40	0.13[a]	LOOK-AT(KEYBOARD)	
19:06.19	19:06.50	0.24	MENTAL	
19:06.74	19:06.86	0.07[a]	TYPE(-)	
19:06.81	19:07.18	0.32[a]	LOOK-AT(MANUSCRIPT)	
19:07.13	19:07.25	0.12	TYPE(e)	
19:07.25	19:07.51	0.26	MENTAL	
19:07.51	19:07.63	0.12	TYPE(x)	
19:07.63	19:09.46	1.83	MENTAL	
19:09.46	19:09.65	0.19	TYPE(< CR >)	
19:09.65	19:09.92	0.27	MENTAL	
19:09.92	19:10.04	0.12	TYPE(e)	
19:10.04	19:10.11	0.07	MENTAL	
19:10.11	19:10.46	0.00[a]	LOOK-AT(CRT)	
19:10.11	19:10.72	0.61	TYPE(x < CR > < CR > /)	
19:10.72	19:11.76	1.04	MENTAL	

[a] Time ΔT charged to operator is less than the difference between the Start and Stop clock times because the operator overlaps with the next operator.

TABLE 4.10

UNIT TASK TIMES FOR S13'S PROTOCOL

	N	Mean (*sec*)	SD (*sec*)
All Unit Tasks:			
Derivation Data	36	13.37	8.87
Cross validation Data	34	19.46	22.97
Total	70	16.33	17.37
Error Unit Tasks:			
Derivation Data	10	16.96	15.13
Crossvalidation Data	15	29.46	32.17
Total	25	24.46	26.99
Error Unit Tasks with Error Time Removed:			
Derivation Data	10	10.69	2.68
Crossvalidation Data	15	13.72	6.47
Total	25	12.51	5.42
Error-Free Unit Tasks:			
Derivation Data	26	11.99	4.55
Crossvalidation Data	19	11.57	3.62
Total	45	11.81	4.14

sisting of the 36 unit tasks on the even-numbered pages. This partition allowed basic operator statistics to be computed on the Deviation set while preserving the Crossvalidation data set for an attempt at prediction in a matched situation where no statistical advantage has been taken of chance. Table 4.10 lays out the gross unit task time statistics for these two data sets.

The data were also partitioned into the set of *error-free* unit tasks and the set of *error* unit tasks, each of the latter containing at least one identifiable error. The criterion for identifying an error is that the user takes some *overt corrective action*, i.e., an action that undoes the effect of a preceding action. The *error time* in an error unit task is the time it takes to perform the corrective action plus any preceding action that is undone by the corrective action. For example, the error time for a mistyped character is the time to type the bad character plus the time to type the control-A (which erases it). Thus error time is the time penalty for error. It can be seen in Table 4.10 that error unit tasks take more

time and are more variable than error-free unit tasks. But, if error time is removed from the error unit tasks, then the remaining (non-error) time in the error unit tasks is comparable to the time for the error-free unit tasks (Mann-Whitney $U(10,26) = 99.0$, $p > 0.10$ for the Derivation data and $U(15,19) = 142.0$, $p > 0.10$ for the Crossvalidation data). Looking at error-free tasks alone, it can also be seen that the set of Derivation unit tasks do not differ significantly in time from the Crossvalidation unit tasks ($U(19,26) = 180.5$, $p > 0.05$). Thus, the two halves of the data are comparable for analysis.

All of the analyses below will use the error-free data. The analysis of errors, while partially within the competence of the GOMS model (most errors being routine) requires a separate analysis (see Card, Moran, and Newell, 1976 for the beginnings of a GOMS analysis of errors).

Fitting the Models to the Data

The protocol record for the error-free Derivation unit tasks was coded into a sequence of operators from each manuscript editing model. For example, the Model M4B coding of the protocol segment in Table 4.10 is:

18:56.33 - 18:59.94	3.61 sec **GET-UT**
18:59.94 - 19:04.16	4.22 sec **USE-QS-METHOD**
19:04.16 - 19:10.72	6.56 sec **USE-S-COMMAND**
19:10.72 - 19:11.23	0.51 sec **VERIFY-EDIT**

To encode each operator requires a recognizer that determines whether the operator occurs in the data and, if so, what its boundary times are. Such recognizers are insensitive to many of the details of what happens. An odd **MENTAL** operator within a **SPECIFY-CMD** (at Level 2), a **USE-QS-METHOD** (at Level 4), or an **EDIT-UT** (at Level 16) is quite consistent and is accepted by the recognizers for these operators. Thus, it is possible—and indeed it is the case—hat the higher-level models account for all of the descriptive operators in the protocol. But these odd descriptive operators (e.g., the odd **MENTAL**) are not without consequence; they may show up as sequence errors and, in chronometric analysis, as variance in the higher-level operator times.

The Level 0.5 models, on the other hand, must map one-to-one onto the protocol, since the Level 0.5 operators are at the same level of aggregation as the protocol operators. Many of the protocol operators (e.g., **TYPE**) are identical to the Level 0.5 operators and are identified directly, whereas other protocol operators (e.g., **MENTAL**) must be relabeled (e.g., as **SEARCH-FOR**, **CHOOSE-CMD**, etc., in Model M0.5E) to fit the models. The possibility then exists that there will be

descriptive operators in the protocol that are not accounted for by the models. More often the descriptive operator, though a possible operator type in the models, may not correspond to any possible operator produced by the models at that point. This happens for 78 out of the 581 operator instances in the protocol. The most significant kind of unaccounted-for operators are instances of **MENTAL** that cannot be interpreted as one of the M0.5E operators; these are labelled **UNKNOWN**. Of the unaccounted-for operators, 71 are **UNKNOWNs**, 6 are **MOVE-HANDs**, and 1 is an **ACTION**. The time for all of the unaccounted-for operators in the protocol amounts to only 8% of the total time and 14% of the operator occurrences. Notice, in this regard, that the **UNKNOWN** operators have an extremely low mean (0.28 sec), many of them simply being brief waits between, or perhaps preparation for, other operators, though there is no way of assigning these in the present models.

It sometimes happens that two mental operators (e.g., **VERIFY-LOC** and **SPECIFY-CMD** in M2A) are predicted by the model to occur in succession. In these cases there is a problem determining the boundary between them, for there is no overt indication in the data. Each operator type involved in such cases (e.g., **VERIFY-LOC**) was compared to instances of the operator where the boundaries were observable (i.e., instances where it was surrounded by non-mental operators). This comparison showed clearly that the operator times of these adjacent mental operators are not additive. These cases are treated as "combined operators," i.e., as if they were separate operator types; and they are given names indicating that they are combined (e.g., **V + SC**). In all there are four different combined operator types, two at Level 2 and two at Level 0.5.

Accuracy of Sequence Predictions

Each of the models was used to predict the sequence of operators in the recoded protocol For method selections, the following rules, simplified from Table 4.8, were used:

Selection rules for ***LOCATE*** *goal:*

R1. Use the **QS-METHOD** as default.
R2. Use the **LF-METHOD** if $D < 5$ lines.

Selection rules for ***MODIFY*** *goal:*

R1. Use the **S-CMD** as default.

In predicting operator sequences it is also necessary to fix the conditions under which the "optional" operators in some of the models would be fired. These operators mainly center around the question of when to invoke extra **GET-FROM-MS** operations, either implicitly (the **/G** versions of the **SPECIFY** operators in models M2B and M2D, see Table 4.4) or explicitly (the **GET-FROM** goal in M0.5E, see Table 4.5). Since the conditions which cause extra **GET-FROM-MS** operations were not clear from the data, each option was decided such that exactly one extra **GET-FROM-MS** was predicted for any unit task.

How well the predicted sequences fit the observed sequences was assessed in two ways. First, the set of frequencies with which each operator was predicted to occur was correlated with the set of frequencies which were observed in S13's data. Second, the operator sequences for each model were transformed into a matrix of transitions between each operator type; and the corresponding matrices were correlated cell by cell.

Both correlations are plotted in Plate 4.1. The points for the different models are joined by lines indicating which models can be derived from which other models by refinement (see Table 4.2). The basic result is that the structural accuracy is extremely high ($r > 0.99$) on both measures for all the models except M2B, M2D, and M0.5E (r between 0.62 and 0.90). These latter models are the ones which required predictions on when to do **GET-FROM-MS** operations. Overall, the correlation between predicted and observed operator frequencies range from .76 to 1.00; between predicted and observed transition frequencies, .62 to 1.00. The method selection rules were 94% accurate overall (one **LOCATE** method and one **MODIFY** method were wrongly predicted in the Derivation data and three Locate Methods were wrongly predicted in the Crossvalidation data). However, method selection errors affect only a small percentage of the operators in the operator sequence, as can be seen in the high correlations of models M4B, M2A, and M2C.

4.32 Task Times

How well can we predict the time it will take to edit the manuscript? The recodings of S13's protocol contain times from which it is possible to compute chronometric statistics for each operator in each model. Estimates of the time to perform a specific unit task were computed in two ways. (1) Given the observed sequence of operators, sum the mean times for each operator in the sequence. This estimate, which we shall call a "reproduction" of the data, gives us an upper bound on how well the models do. (2) Use the sequence of operators predicted by the

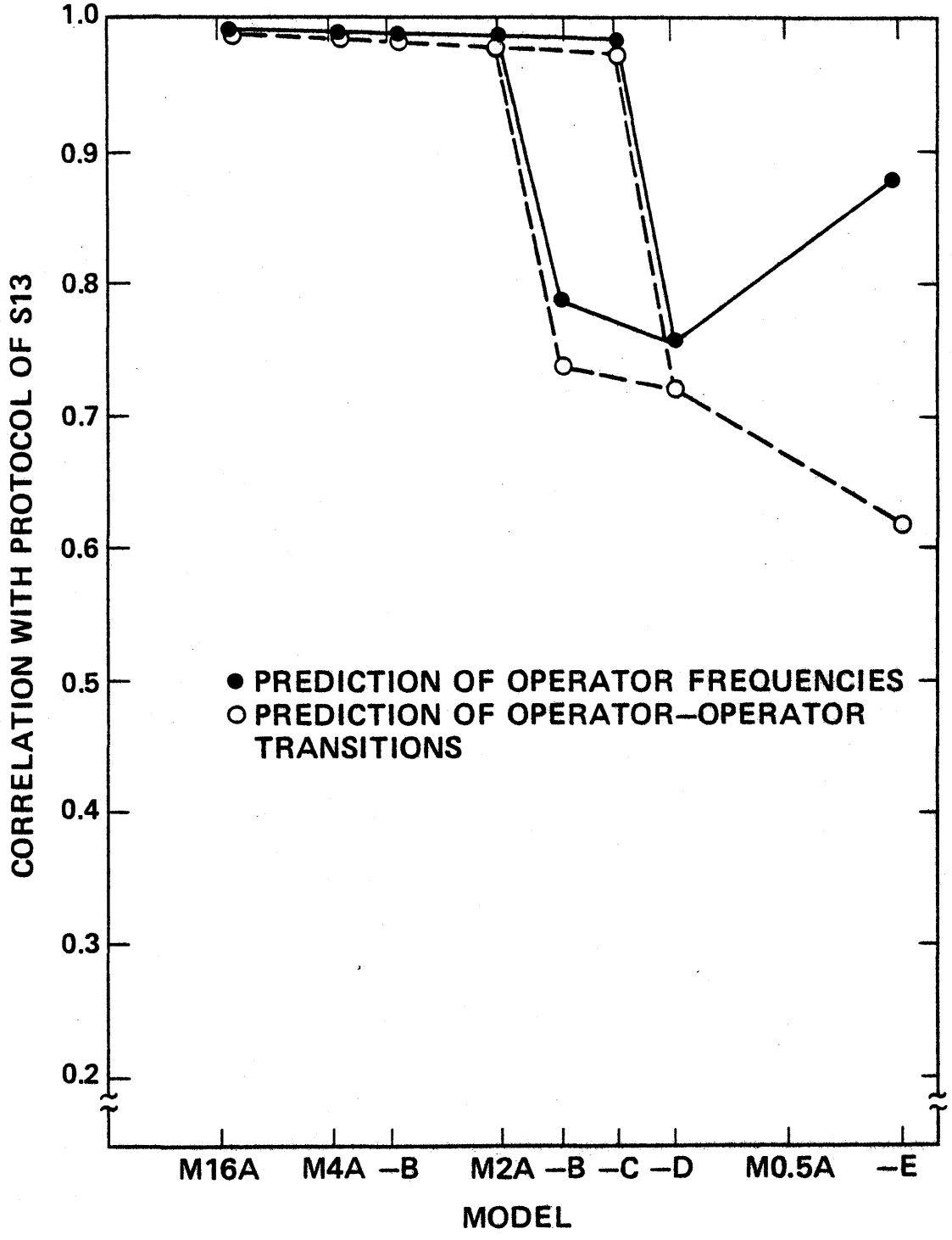


Plate 4.1. Accuracy of operator sequence prediction by models.

models, summing the mean times for each operator in the sequence as before. This estimate, which we shall call a "prediction," should correspond with what we might expect to find applying the models in practice. Error can enter into the estimate either because an operator actually takes longer in some contexts than others, as the operator **SPECIFY-ARG** takes longer when specifying an argument to the Modify command in POET than to the Substitute command; or error can enter because the model predicts the wrong sequence of operators and the wrong sequence is predicted to take a different amount of time than the correct sequence actually requires. Comparing the prediction result with the result for the reproduction of the data gives us a way to assess where the model's most important sources of error lie.

Estimation of Operator Times

The durations of all operator instances of each operator type in the Derivation data were used to estimate the operator times. Table 4.11 is a table of the empirically determined duration for each operator of each model derived from the Derivation data. This table presents the operators in the order they are described in the manuscript editing models in Tables 4.3 to 4.5. The operators for Models M16A, M4A, M4B, M2A, and M0.5A are given in their entirety, so that some operators occur more than once. Only the operators unique to that level are given for the other models at Level 2 and Level 0.5.

Since the data come from a naturalistic situation and since a rare method may appear only once in the data, there is a fair chance that some radically extreme times may show up in the distributions of operator times. Though these must be accepted in any prediction test, it is appropriate to avoid them in estimating the characteristics of the operators. Consequently, in Table 4.11 outliers that lie beyond two SDs from the raw mean have been dropped and the mean, SD, and CV recomputed for each operator. The number of high and low outliers dropped are indicated in the last column of the table.

All operators except one are modeled as taking constant time. The exception is the **TYPE** operator. While it is obvious that **TYPE** should be parameterized by the number of characters to be typed, we must be able to predict what search strings and what substitution strings will be used in order to capitalize on the parameterization. At the outset, it is unknown whether inaccuracies in predicting the string to be used will wipe out any gains from using a more accurate model. To find out, model M05E' was defined to be like Model M05E in every way except that it uses constant 0.39 sec, the average time per typing burst, as the time for **TYPE**. For Models M05A and M05E the time for **TYPE** is

TABLE 4.11

OPERATOR STATISTICS FOR ALL MODELS

Operator	Mean (sec)	SD (sec)	CV	%Time	N	%N	L,H
Model M16A							
EDIT-UT	11.38	3.36	0.30	100%	26	100%	0,1
Model M4A							
GET-NEXT-PAGE	2.14	1.37	0.64	3%	5	5%	0,0
GET-UT	1.92	0.64	0.33	16%	24	23%	0,1
LOCATE-LINE	3.98	1.16	0.29	32%	24	23%	0,1
MODIFY-TEXT	3.85	1.54	0.40	35%	26	25%	0,1
VERIFY-EDIT	1.49	0.85	0.57	14%	26	25%	0,1
Average	2.77	1.07	0.41				
Model M4B							
GET-NEXT-PAGE	2.14	1.37	0.64	3%	5	5%	0,0
GET-UT	1.92	0.64	0.33	16%	24	23%	0,1
USE-QS-METHOD	3.94	1.19	0.30	28%	21	20%	0,1
USE-LF-METHOD	4.27	1.05	0.25	4%	3	3%	0,0
USE-S-CMD	3.63	1.36	0.37	29%	24	23%	0,1
USE-M-CMD	9.72	6.10	0.63	6%	2	2%	0,0
VERIFY-EDIT	1.49	0.85	0.57	14%	26	25%	0,1
Average	2.83	1.12	0.41				

NOTE: Column N gives the total number of occurrences of each operator type in each model. Column %N gives the percentage of occurrences and Column %T gives the percentage of total time of each operator type in each model. An initial set of statistics was computed for each operator type, and all instances lying more than two SDs from the mean were declared as outliers. Column "L,H" gives the number of low and high outliers. After discarding the outliers, the statistics for each operator type were recomputed. These are given in Columns Mean, SD, and CV (CV = SD/Mean). The rows labelled "Average" give the N-weighted average statistics for all operators in each model.

TABLE 4.11 (continued)

Operator	Mean (sec)	SD (sec)	CV	%Time	N	%N	L,H
Model M2A							
GET-NEXT-PAGE	2.14	1.37	0.64	3%	5	3%	0,0
GET-FROM-MS	2.06	0.91	0.44	4%	6	3%	0,0
GFM + SC	1.80	0.40	0.22	12%	18	10%	0,1
VERIFY-LOC	1.94	0.87	0.45	12%	17	9%	0,1
V + SC	2.00	0.88	0.44	4%	7	4%	0,0
VERIFY-EDIT	1.49	0.85	0.57	14%	26	14%	0,1
SPECIFY-CMD	1.47	1.14	0.77	13%	28	15%	0,0
SPECIFY-ARG	1.46	0.84	0.57	38%	76	42%	0,3
Average	1.60	0.84	0.55				
Model M2B	*The operators in M2B are the same as in M2A except for* ***SPECIFY-CMD*** *and* ***SPECIFY-ARG****, which are expanded below.*						
SPECIFY-CMD/NG	0.40	0.35	0.88	2%	11	6%	0,1
SPECIFY-CMD/G	2.03	0.99	0.49	11%	17	9%	0,0
SPECIFY-ARG/NG	1.29	0.70	0.54	29%	63	34%	0,3
SPECIFY-ARG/G	2.28	1.02	0.45	10%	13	7%	0,0
Average	1.59	0.76	0.51				
Model M2C	*The operators in M2C are the same as in M2A except for* ***SPECIFY-ARG****, which is expanded below.*						
SPECIFY-ARG/Q	2.07	0.57	0.28	14%	21	11%	1,1
SPECIFY-ARG/S1	1.34	0.94	0.70	12%	24	13%	0,1
SPECIFY-ARG/S2	0.94	0.29	0.31	8%	24	13%	0,2
SPECIFY-ARG/M	2.04	1.36	0.67	5%	7	4%	0,0
Average	1.61	0.80	0.50				
Model M2D	*The operators of M2D are the same as in M2B expect for* ***SPECIFY-ARG****, which is expanded below.*						
SPECIFY-ARG/Q/NG	1.94	0.42	0.22	9%	14	8%	1,1
SPECIFY-ARG/Q/G	2.29	0.75	0.33	5%	7	4%	0,0
SPECIFY-ARG/S1/NG	1.12	0.73	0.65	9%	21	11%	0,1
SPECIFY-ARG/S1/G	2.79	0.96	0.34	3%	3	2%	0,0
SPECIFY-ARG/S2/NG	0.93	0.29	0.32	8%	23	13%	0,2
SPECIFY-ARG/S2/G	1.20	-	-	0%	1	1%	0,0
SPECIFY-ARG/M/NG	2.05	1.21	0.59	3%	5	3%	0,0
SPECIFY-ARG/M/G	2.02	2.29	1.13	1%	2	1%	0,0
Average	1.60	0.70	0.47				

TABLE 4.11 (continued)

Operator	Mean (sec)	SD (sec)	CV	%Time	N	%N	L,H
Model M0.5A:							
MENTAL	0.62	0.54	0.88	60%	260	43%	0,12
TYPE	0.39[a]	0.12[a]	0.31[a]	22%	173	28%	-
LOOK-AT	0.31	0.10	0.32	13%	139	23%	9,1
HOME	0.52	0.11	0.22	2%	9	1%	0,1
TURN-PAGE	0.67	0.21	0.32	1%	5	1%	0,0
MOVE-HAND	0.19	0.17	0.91	1%	17	3%	0,1
ACTION	0.13	0.19	1.56	0%	6	1%	0,0
EXPRESSION	0.23	-	-	0%	1	0%	0,0
Average	0.47	0.30	0.58				
Model M0.5E *The operators for M0.5E are the same as in M0.5A except for* ***MENTAL****, which is expanded below.*							
SEARCH-FOR	0.72	0.51	0.71	7%	28	5%	0,1
SF + CM	1.07	0.56	0.52	7%	20	3%	0,0
CHOOSE-CMD	0.74	0.42	0.57	2%	8	1%	0,0
CHOOSE-ARG	0.41	0.33	0.81	9%	56	9%	0,4
COMPARE	1.01	0.83	0.82	22%	59	10%	0,3
C + CM	1.14	0.68	0.60	7%	18	3%	0,0
UNKNOWN	0.28	0.25	0.92	8%	71	12%	0,2
Average	0.48	0.29	0.55				

[a] Since **TYPE** is a parameterized operator, the number given in the SD column is the Standard Error of the parameterized typing prediction, given in the equation in Section 4.31. The Mean for **TYPE** is the mean of all observed **TYPE** operations, i.e., as if **TYPE** were a constant operator. The CV given for **TYPE** is SE/Mean.

parameterized by the number of shift characters N_{shift}, carriage returns N_{cr}, and other characters N_{other} according to the equation

$$T = 0.05 + 0.17N_{shift} + 0.19N_{cr} + 0.11N_{other} \text{ sec.} \quad (1)$$

The equation is based on the regression fit of 157 short typing bursts from the experiment (1 to 18 characters in a burst, mean 3.8 characters). This model predicts the typing time rather well ($R^2 = 0.92$, all coefficients significantly different from 0 at $p \ll 10^{-4}$), and it is somewhat better than the simpler model:

$$T = 0.06 + 0.12N_{char} \text{ sec}$$

($R^2 = 0.89$). That S13 is a fast typist is apparent from these equations (0.12 sec per character = 91 words per minute).

Accuracy of Time Predictions

Plate 4.2 presents the results of reproducing the Derivation data with each the models (solid dots). Reproduction of the Derivation data is the weakest prediction because it derives the estimates of operator times from the same data that is being predicted and because the operator sequence is taken as given and not predicted by the independently derived selection rules. It is useful because it gives an upper bound on how well the models can be expected to predict other data. The accuracy of the reproduction is evaluated by comparing predicted unit task times with the user's actual unit task times. The average prediction error for each unit task is summarized by the root-mean-square of the prediction errors (RMSE)[1] expressed as a percentage of the average actual unit task time. The reproduction accuracy improves as the models become more detailed, but the rate of improvement diminishes. The average prediction error (RMSE) is about 40% when using the average unit task time as the predictor (Model M16A), and this is cut to 20% by using the most detailed model (M0.5E).

Plate 4.2 also presents the results of using the operator sequences predicted by the models as the basis for predictions of the times per unit task for both the Derivation and the Crossvalidation data sets. M0.5E was not applied to the Derivation data because, given our intimate familiarity with that data, we could not fairly a priori predict the **TYPE** strings.

The main result is that the models except M16A on the Derivation data are all about equally accurate at prediction, with an RMSE of about 30%. A study of the prediction errors on unit tasks with different task environment features revealed that the only task environment feature that

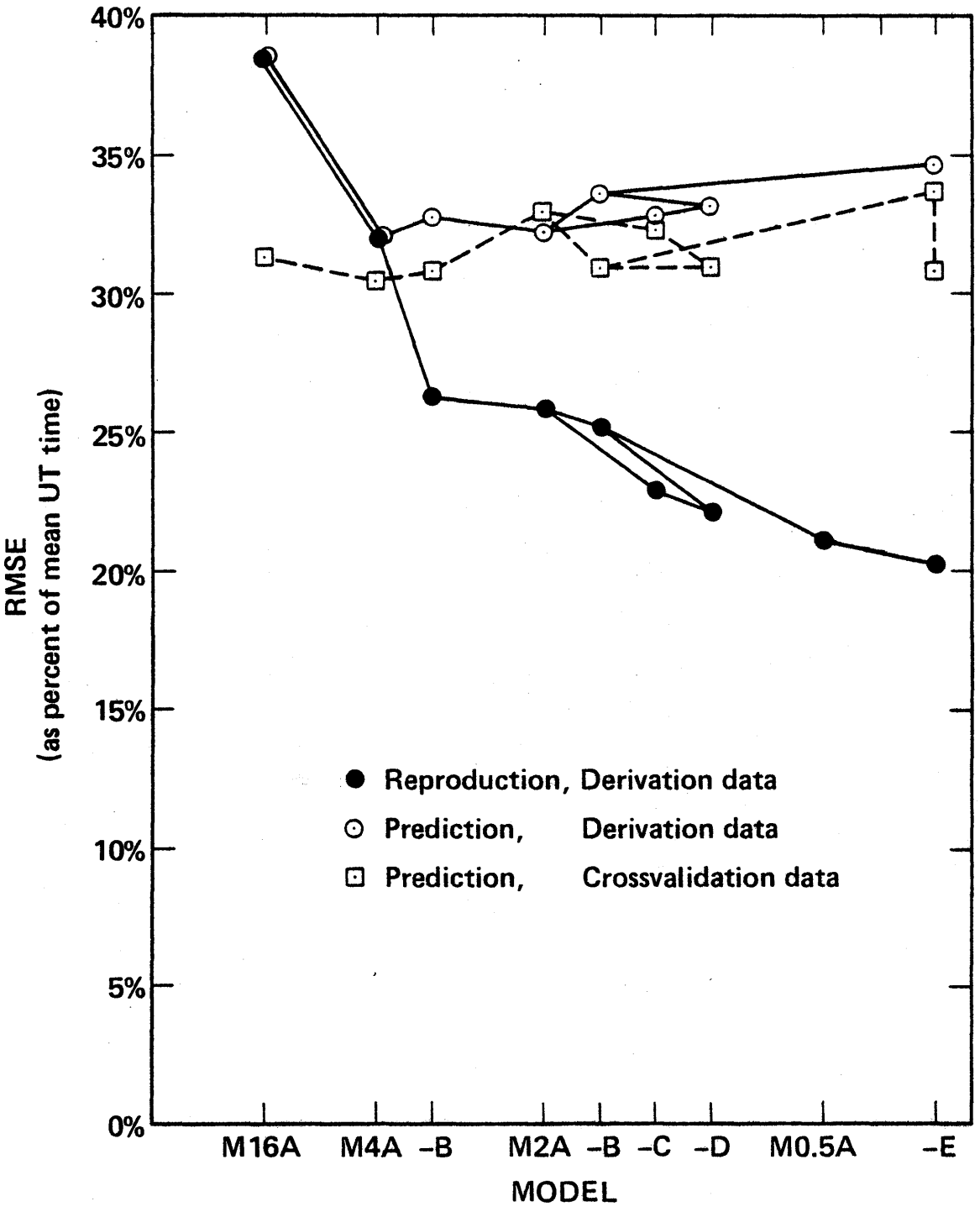


Plate 4.2. Accuracy of unit task time prediction by models.

allowed any prediction gain was when the unit task shared the same line with another unit task. There are two tasks with the this feature in the Derivation data, and they are the sole reason why M16A predicts more poorly on that data. Conversely, the lack of such features in any of the Crossvalidation unit tasks allows M16A to do better on that data.

The prediction of the unit task times of the data is slightly less accurate than the reproduction of these times. Since the predicted time per operator is the same in both cases, the difference is due to the difference between the predicted and the actual operator sequences.

If the RMSE measure is interpreted as the average prediction error, the 20% to 40% range of the models may seem to be high. But predicting editing times unit task by unit task for a single user is a very stringent test. If the unit of prediction were the whole manuscript rather than the unit task, then the prediction error would drop considerably, since the high and low predictions of the various unit tasks would tend to cancel each other. The RMSE approximately obeys a square root of n law, where n is the number of unit tasks[1]. Thus the RMSE for predicting the time to edit the whole manuscript with the reproduction models would range from $(40\%)(64)^{-1/2} = 5\%$ for the worst model to 2.5% for the best. The RMSE for the prediction models would be around 3-4%. The error for these models of variable-sequence, cognitive activity would thus seem to be in the same range ($\sim$ 5%) as that often cited for pre-determined time system predictions of invariant-sequence, physical activity by industrial engineers (Maynard, 1971).

4.4 DISCUSSION

Assessment of the Models

Description of Behavior. From the case study of S13's protocol, it is apparent that descriptions of a user's behavior in the manuscript editing task can be constructed from a reasonably small number of components. Depending on the grain of analysis, the behavior could be described by 1-20 goals, 1-13 operators, 4-6 methods, and 2-4 selection rules. Moreover, this description is a reasonably accurate account of a user's behavior in the task. The selection rules were able to predict the users choice of methods about 90% of the time. The frequency mix of different operators predicted by the model correlated well (r = 0.76 to 1.00) with the observed mix. The operator to operator transition matrix predicted by the models matched well (r = 0.62 to 1.00) the transition matrices calculated from the data. Thus, it would appear that the behavior sequence of a user in a naturalistic text-editing situation can, in

fact, be described as repetitions of patterns built from a handful of components by the GOMS models.

Prediction of Task Times. The GOMS models likewise provide a reasonable prediction for the amount of time a task consumes. The models were able to predict the time per unit task to within about 30% on new (Crossvalidation) data. In this situation, the models had to predict the times for all the operators as well as the operator sequence. Predicting the unit task times to within 30% is equivalent to predicting the time to edit the whole manuscript to within about 4%, an accuracy comparable to that achieved by industrial engineering "predetermined time" systems for repetitive manual operations.

Grain of Analysis. How does the ability of the GOMS models to predict the behavior of the user vary as a function of the grain of analysis? The short answer is: When the sequence of operators is given, refinement of the grain size improves the ability to predict performance times. But when the sequence of operators must be predicted, refinement of the grain size causes the ability to predict the sequence of the refined operators to deteriorate (Plate 4.1); ability to predict performance times does not change (Plate 4.2).

Because of the very large amounts of data analysis required, it was not feasible to compute directly the reproduction of the Crossvalidation data. The small differences found between the results of the prediction on the Derivation data and on the Crossvalidation data, however, suggest that the reproduction of the Crossvalidation data would parallel those for the Derivation data and that the improvement of time accuracy with refinement of analysis grain is not solely attributable to capitalization on chance.

If the models could predict operator sequences perfectly, then the prediction curves in Plate 4.2 would drop to the reproduction curve. That the prediction curves are essentially horizontal implies that refining the grain of analysis did not tap the sources of time variability. In the models, variability is expressed in the method selection rules and optionality conditions, which are triggered by features of the task environment. Thus, either the models didn't capitalize on all the available features in the task environment, or there were no task environment features that gave clues to the variability.

In fact, it is important to note that the variability in the set of unit tasks in the experiment is quite small, both with respect to the user's performance times (see Table 4.10) and with respect to the range of edit tasks on the manuscript—all are small edits of about the same complexity. This low variance was intentional, since we were not trying to manipulate the task environment, but were trying to assess the natural variability in the data and the ability of various models to deal with it.

It appears that while the models, as a whole, were not bad at predicting the average time per unit task, there was insufficient variation within the editing tasks to trigger increased responsiveness from the finer grain models.

The level of variability for which the models *can* account begins to show up in the difference between M16A and M4A in predicting the Derivation data and between M16A, M4A, and M4B in the reproductions. The models do improve in performance down to about Level 4. Thus some variability is expressible in terms of combinations of Level 4 operators.

General Issues of Model Structure

What psychological reality is to be ascribed to the various components and features of the GOMS model?

Goals. The occurrence of goals in the GOMS model is one of its primary cognitive features. Goals are required in generating the model and in supporting its rational character as behavior directed towards the end of editing the manuscript. As it stands, however, the goals do not make any distinguishable contribution to the timing calculations of the various models. Technically, this arises from a confounding of goals and methods/operators, so that any time assigned to creating a goal or to cleaning up and disposing of a goal would not be distinguishable from additional time in the associated operators. Goal-manipulation operations should not take longer than about 0.5 sec, so that goal operators should not show up at the Level 2 or above in any event.

The confounding of goal manipulation times results in part from GOMS being a model of skilled behavior, so that the overt record contains evidence only of the sequence of effective actions. This can be confirmed from the verbal expressions made during manuscript editing. In our user, there are no verbal expressions that indicate goal activity. However, protocols from inexperienced users are sprinkled with goal statements that correspond to the goals in the GOMS model. In one such experiment, when the model predicted the processing of **GOAL: USE QS-METHOD**, the user would almost invariably make comments like: "Okay, I want to get down to a line that starts with 'Food store' ". When the model predicted **GOAL: USE S-CMD**, the user would say: "Now I want to substitute '30' for '39' ". But in line with this view no verbalizations occurred related to operator actions like **TYPE**.

Uniformity of Level. Taking a model at a given grain size, like 2 sec, tends to create a set of operators which are homogeneous in their duration. Operators much larger than the grain size are always decomposed. Operators much smaller than the grain sizes can be

discarded on grounds of making only insubstantial contribution to the total time.

From a practical point of view, it might be just as effective to use a single operator time for all operators at a given grain size. There is some indication (Wainer, 1976; Claudy, 1972) that such a procedure would not materially reduce the prediction accuracy, while increasing the robustness of the estimate and enhancing the use of such models in applied work.

Operator Variability. The order of precision of our operators, as measured by the coefficient of variation (CV = the standard deviation/mean), ranges from about 0.9 at Level 0.5 to 0.3 at Level 16. In general, CVs should be expected to decrease with increasing mean in situations with compositions of sums of elements. The actual decrease is illustrated in Plate 4.3, which plots CV against operator mean (M). The open symbols represent operators from Table 4.11 that occurred more than 5 times in S13's protocol. The other symbols on the graph are from measurements of industrial operations in which there is some significant cognitive element. The solid circles are operators in a ladies garment factory, such as cutting and stitching clothing patterns, from Abruzzi (1956, pp. 222 & 235); the crosses are from a study by Mills and Hatfield (1974) of a data lookup and entry task. In log-log coordinates the relationship between mean and CV is essentially linear. In fact, a regression analysis of the data in Plate 4.3 gives the following equation:

$$\log CV = -0.340 - 0.211 \log M$$

or

$$CV = 0.457M^{-0.211}$$

This equation explains 58% of the variance, and the coefficient -0.211 is very significantly different from zero ($t(82) = -10.74$, $p \ll 10^{-4}$). Plate 4.3 suggests that in absolute terms the CVs observed in our experiment are roughly what would be expected from the size of the operations alone.

As CV increases, the number of observations needed to estimate the mean to a fixed precision also increases (see Abruzzi, 1956). This is reflected in the figure as greater dispersion of the points about the line as for small M, and in the fact that many of the points on the outlying edge are those with the lowest Ns.

The dispersion of the points in Plate 4.3 also reflects the tendency of physical operators to have lower CVs than mental operators of the same mean. In the figure nearly all of the purely mental operators (indicated by triangles) lie above the regression line while the purely physical

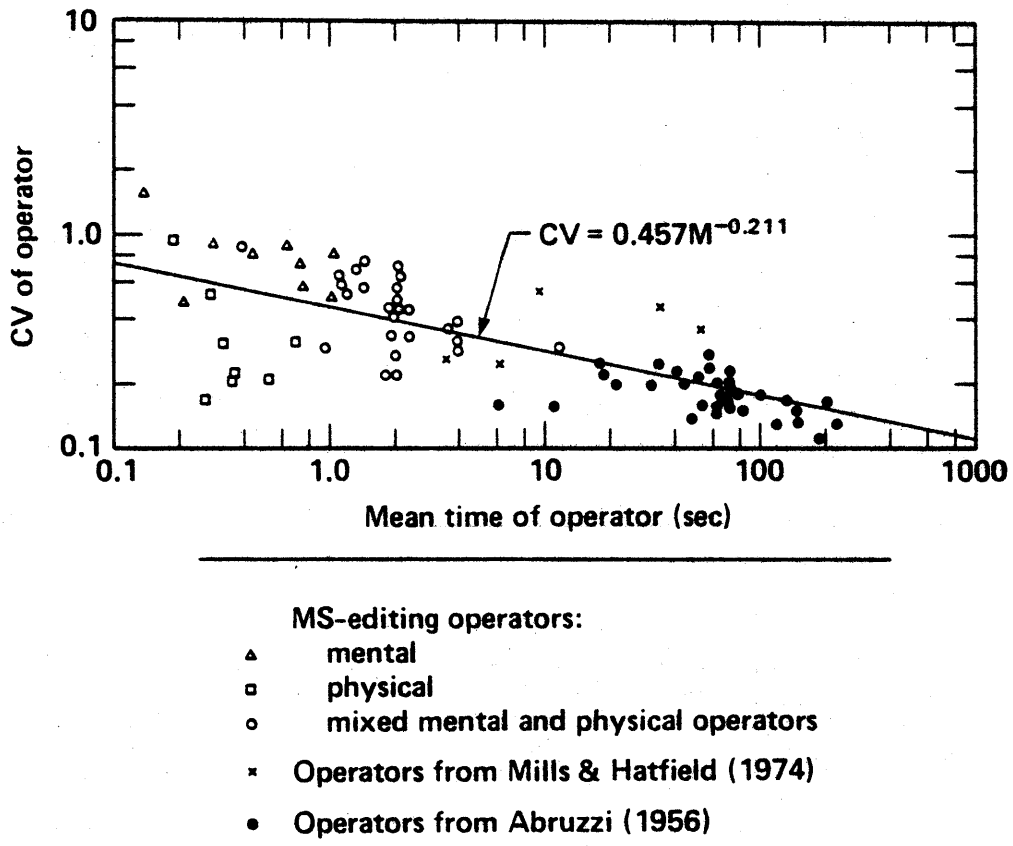


Plate 4.3. Variability of operators.

operators (indicated by squares) lie below. The outlying points below the regression line are simple physical acts such as **LOOK** and **TURN-PAGE**. The outlying points above the line are mental actions such as **CHOOSE-CMD** or **VERIFY-EDIT**. As the time for the operators becomes shorter, approaching the grain of characteristic physiological events, the operators tend to become more purely physical or mental. Since the physical operators are easier to identify and measure, these should have lower *CV*s.

4.5 CONCLUSION

In this chapter we have sought to refine our understanding of the manuscript editing task by examining in detail the parts of the modification cycle. We have examined one editor (POET) in its natural setting, proposed a theory for it (GOMS), and used that theory to describe the behavior. We have used the theory to perform various reproductions and predictions, to assess the stability of the constructs estimated in the theory and to identify sources of naturally occurring variability. We have restricted our investigation to a single subject and a single session so that we could carry out an intensive analysis of the data, but our results are supported in general by other subjects performing manuscript editing tasks in our laboratory.

We can now proceed to give answers to our original questions.

(1) *Can the behavior of a user in a manuscript editing task be described as the combination of a small number of elementary behavioral acts?* Yes. The error-free behavior in manuscript editing is satisfactorily described by the GOMS model. This description is built up from a small number of operators, goal types, methods, and selection rules; and these components reflect the structure of the manuscript editing task environment. The number of operators required depends on the grain of the model but was never more than about 16. We have shown how the model can be extended to handle most error-handling behavior.

(2) *Can we predict the stream of these acts from an analysis of the environment?* Use of the model to predict new but similar data (zero-parameter prediction) reveals that the sources of variability are difficult to tap. Models at all levels yield about the same quality of predictions to new

data. The relatively low variance in the data makes the results not surprising, though somewhat disappointing. Viewed negatively, these results say that measurement in an essentially identical situation is as good a predictor as the theory; viewed positively, these say that predictions can be made from component operators at any level, even if direct measurements are unavailable. Even in the harshest prediction's test, the first order refinement (M4) was superior to the constant time per task model used in Chapter 2.

(3) *Finally, what grain size is appropriate for modeling?* The behavior can be satisfactorily described at several levels (time grains), which constitute consistent decompositions within the GOMS model. No really preferred level of description was found, reflecting in part that the user's behavior is organized at all of these levels, as the GOMS model asserts. Models increase in generality (task independence) as level decreases; they increase modestly in descriptive power; but they become more expensive to construct.

Probably the most important feature of the manuscript editing task to emerge is it's unit task structure, which sets it apart from other tasks (typing, reading, tracking) that have received more attention. The practical usefulness of the unit task concept will be demonstrated in the next chapter.

NOTES

[1] RMSE = $[\Sigma E^2/(n\text{-}1)]^{1/2}$, where E is the distribution of prediction errors over the unit tasks. RMSE is the standard deviation of E about zero, instead of the actual mean of E, and thus RMSE $\geq$ SD (E). If Mean(E) = 0, then RMSE = SD(E), and RMSE is equivalent to the standard error. The calculation should actually be done with SD(E) about Mean(E), but the use of the RMSE is approximately correct if Mean(E) is close to zero.

[2] In attempting to build a statistical model to partition the variance for each model into the operator variance and the structural variance (i.e., the conditionality of the operator sequence), some of the variance remains unaccounted for. Part of the difficulty, apparently, is the existence of covariances between *non-adjacent* operators, even in the Level 4 models. Hence, we must reserve judgement about the independence of the operators.

[3] We would like to acknowledge George Baylor, from the University of Montreal, for helping us formulate the method selection issue and for running some pilot experiments.

5

APPLICATION: Predicting User Performance at an Early Stage in System Design†

In this chapter we apply some of the principles of Chapter 4 to a typical industrial systems design problem.

†Adapted from an unpublished memorandum by Card, Moran, and Newell.

5.1 STATEMENT OF THE PROBLEM

A computer-based system is to be built in which it will be possible for a user to do the layout for a journal roughly in the style of *Cognitive Psychology*. The user, starting from separate on-line files of main text, figures, captions and footnotes has the task of arranging all of these elements into a final page. This job is called "formating" and it includes as well the rendering of text words into different fonts; the treatment of headings; and the final numbering of figures, cross-reference pages, and footnotes (see Figure 5.1). The problem is to make a rough order-of-magnitude estimate of the time per page a user would be expected to spend formatting. Since the final command dialogue has not yet been designed, the estimate can not depend on its details.

5.2 UNIT TASKS SOLUTION

The problem can be solved by recalling from Chapter 4 that the user's behavior is organized into "unit task cycles." By estimating the number of unit tasks necessary to accomplish the formating of a typical page and multiplying that number by the estimated time per unit task, the formating time/page can be estimated.

Step 1. Analyze the problem into unit tasks.

Chapter 4 found that a user's behavior in editing is organized into a sequence of unit task cycles of the same basic operations. Taking into account that the user may have to await the response of the system before he can proceed, the cycle can be written

GET-NEXT-TASK	**(G)**
LOCATE-TASK-ELEMENTS	**(L)**
MAKE-MODIFICATION	**(M)**
WAIT-FOR-SYSTEM (*Optional*)	**(W)**
VERIFY	**(V)**

A typical subtask the user must perform is to specify to the system the type font of a section heading. We do not know the exact commands he will use since the system has yet to be specified, but it can be expected that the necessary actions will fall into a unit task pattern:

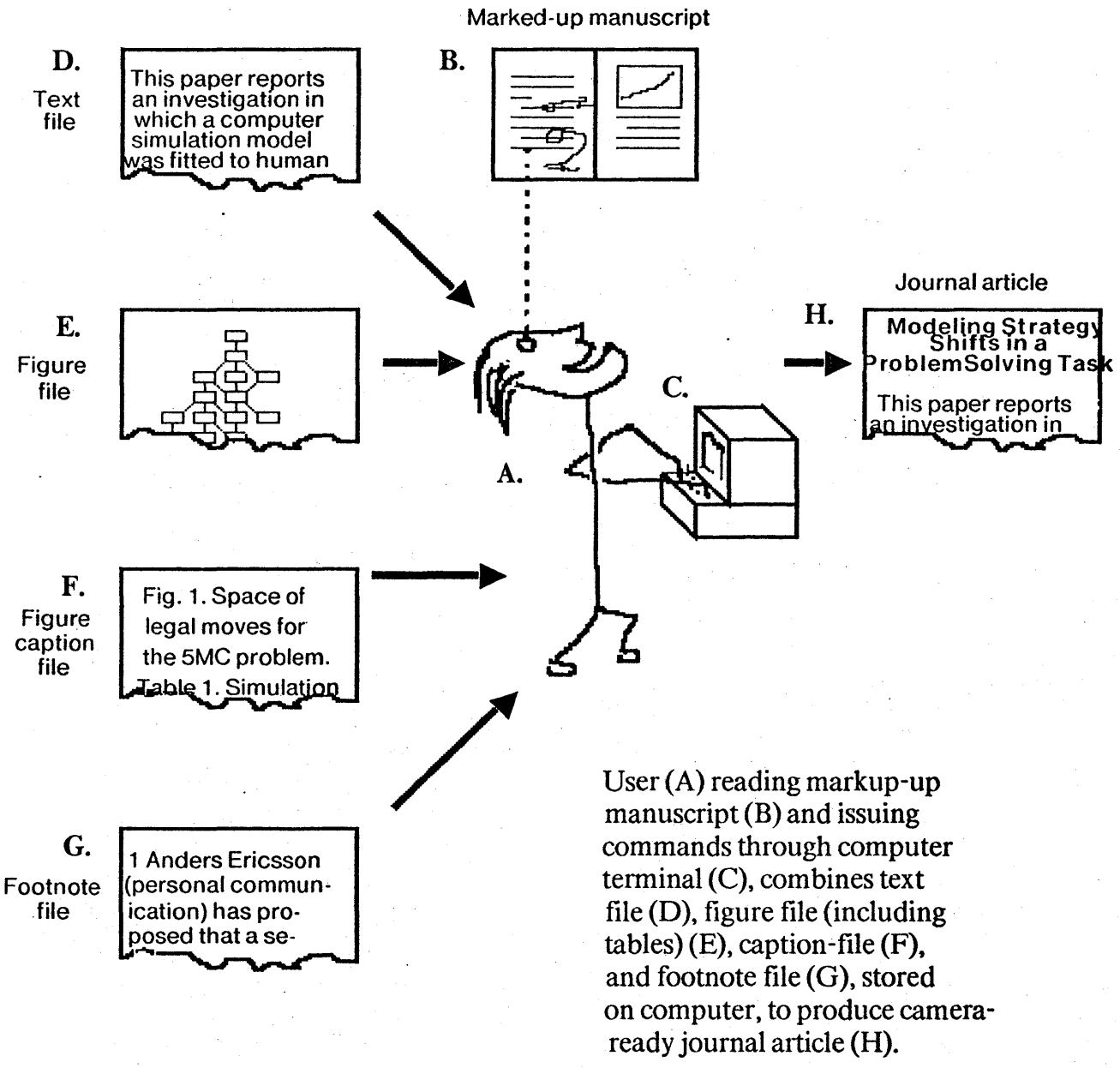


Figure 5.1. Conceptual design of problem system

Specify heading font:	
Detect heading	**G**
Point to entire heading	**L**
Change font of heading	**M**
Verify change	**V**

Thus, the first step in solving the problem is to analyze the task environment into these unit tasks. The unit tasks in the current problem have been entered onto Form A in Figure 5.2. There are 34 unit tasks centered around starting a new page, headings, the text body, special treatments of text words, figures, captions, and footnotes.

Step 2. Identify a set of Macro Operations.

Often the user will be performing a number of related tasks in sequence. For every heading he inserts into the text he will need to specify the font in which the heading is to be rendered, to specify the vertical placement, to specify the horizontal placement, and to specify the heading number. We can simplify the analysis by identifying a *macro operation, Process Heading* (abbreviated *#Ph*). As macro operations are identified they are written in the heavy boxes of Form B (Figure 5.3). For the current problem, five macro operations can be identified: *Process new page* (abbreviated *#Pp*), *Process heading* (*#Ph*), *Process figure* (*#Pf*), *Process footnote* (*#Pf*), and *Process reference* (*#Pr*).

5.22 Analysis of System

Step 3 List unit task components of the macro operations, recording system assumptions.

In order to proceed further, it is necessary to specify some broad characteristics of the system. Will it be necessary for the user manually to number each page? Need he manually indent each paragraph? Even though a system is in the pre-specification stage, it may be possible to settle on these general issues. If not, then alternative systems can be specified so as to examine the range in behavior induced by the alternative designs. The latter course will be followed here. We shall analyze two systems for the problem: System A, in which the user must do a great deal by hand, and System B, which is more automatic.

FORM A Enumeration of Unit Tasks		
ID	Description	Wait
Lp	Load page	X
Vp	Specify vertical extent of page	
Hp	Specify horizontal extent of page	
Np	Specify page number	
Lh	Load heading	X
Vh	Specify vertical spacing of heading	
Hh	Specify horozontal placement of heading	
Fh	Specify heading font	
Nh	Specify heading number	
Lhk	Load contant heading	X
Vhk	Specify vertical spacing of constant heading	
Hhk	Specify horozontal placement of constant heading	
Fhk	Specify constant heading font	
Lb	Load body of textual material	X
Vb	Specify vertical placement of text body	
Hb	Specify horozontal placement of text body (e.g. indent)	
Fb	Specify font of text body	
Nb	Specify numbering of text body	
Lt	Load text word	X
Ft	Specify font of text word	
Nt	Specify number for text word (e.g., footnote number)	
Lf	Load figure	X
Vf	Specify vertical placement of figure	X
Hf	Specify horozontal placement of figure	X
Lc	Load caption	X
Vc	Specify vertical placement of caption	X
Hc	Specify horoztonal placement of caption	X
Fc	Specify font of caption	
Nc	Specify figure number (which goes with caption)	
Ln	Load footnote	X
Vn	Specify vertical placement of footnote	X
Hn	Specify horozontal placement of footnote	X
Fn	Specify footnote font	
Nn	Specify footnote number (with note body, not callout in text)	

Figure 5.2. Unit tasks for problem

FORM B
Macro Operations

Page 1 of 2

System: A

	ID	Description	Waits	UTs
MACRO OP	# Pp	Process new page	1	2
COMPONENT UNIT TASKS	Lp	Load page	X	
	Np	Specify page number		

MACRO OP	# Ph	Process heading	0	4
COMPONENT UNIT TASKS	Fh	Specify font of heading		
	Vh	Specify vertical spacing of heading		
	Hh	Specify horizontal placment of heading		
	Nh	Specify heading number		

MACRO OP	# Pf	Process figure and caption	6	9
COMPONENT UNIT TASKS	Lf	Load figure	X	
	Vf	Specify vertical location of figure	X	
	Hf	Specify horiztonal location of figure	X	
	Lc	Load caption	X	
	Vc	Specify vertical location of caption	X	
	Hc	Specify horizontal location of caption	X	
	Fc	Specify font of caption		
	Nc	Specify number of caption		
	Nt	Number figure callout in text		

Figure 5.3. Macro operations for System A

FORM B
Macro Operations

Page 2 of 2

System: _____

	ID	Description	Waits	UTs
MACRO OP	# Pn	Process footnote	3	6
COMPONENT UNIT TASKS	Ln	Load footnote	X	
	Vn	Specify vertical location of footnote	X	
	Hn	Specify horizontal location of footnote	X	
	Fn	Specify footnote font		
	Nn	Number footnote body		
	Nt	Number footnote callout		

MACRO OP	# Pr	Process reference	0	2
COMPONENT UNIT TASKS	Ft	Italicize journal name		
	Ft	Render volume number in boldface		

MACRO OP				
COMPONENT UNIT TASKS				

Figure 5.3. (continued)

System A will leave to the user the loading of a page and the processing of headings, figures, footnotes, and miscellaneous numbering and word changes. In System B, more work will be taken over by the system.

The analyst proceeds as follows: for each macro operation on Form B, he fills out the unit tasks (from Form A) of which it is composed. This activity is naturally accompanied by decisions and assumptions about the characteristics of the subject system. As each decision or assumption is reached it is entered on Form C (Figure 5.4).

The first macro operation in Figure 5.3 is *Process new page.* In System A, the user must specify the page number himself. So there are two component unit tasks for process page: *Load page* and *Specify page number.* But in System B, the system takes care of page numbering automatically, so only Load page is entered. A note about the automatic page numbering is made on Form C.

Step 4. Compute the number of unit tasks/operation

Using the above assumptions regarding the system, we can count the number of unit tasks from the list in Figure 5.1 necessary to perform each macro operation: *#Pp* (*Process new page*) requires 2 unit tasks, *#Ph* (*Process Heading*) 4, *#Pf* (*Process figure*) 8, and *#Pn* (*Process footnote*) 6.

Step 5. Compute the number of waits/operation

Examination of the unit tasks on Form A, keeping in mind the sort of technology to be used for system implementation, will suggest which unit tasks should have **WAIT** operators attached to them. *Lp* (*Load page*) will likely be one unit task that will require the user to wait while a new page is loaded in and displayed on the screen, whereas it is expected that *Np* (*Specify page number*) will be executed without noticeable delay. Those unit tasks in Form A for which some wait by the user is expected have been so indicated.

Step 6. Estimate the time/unit task and time/wait

A reasonable assumption for the time/unit task required by users of System A is that it would be in the range of the two display-based editors measured in Chapter 2. Combining the time/unit task in Table 2.2 for correct tasks with Editor Y and RCG yields (10.1 + 7.2)/2 = 8.65, which we round to 8.5 sec/unit task.

FORM C
System Characterization

System: A

Item	Description
A1	Typed headings embedded in text (no Lh)
A2	Text loaded on call (Lt) with page size, margins, position, and font automatic (no Vp, Hp, Vb, Hb, Fb)
A3	Paragraphs need to be indented manually
A4	Either one or two figures fit together horizontally
A5	Text cannot go horizontally beside figure
A6	Figure can go vertically either at top, at bottom, or in middle
A7	When figure is moved, overlaid text automatically slides around, possibly onto next page
A8	All headings must be set manually
A9	All numbers (Nt) are kept track of by the system, but must be placed by user

Time/UT = 8.5	Time/Wait = 6.0

Figure 5.4. Characterization of System A

In order to set a number for the time/wait, a number of exercises were performed, using various display-based systems, in which the user would have to wait for a system response. The time to load a file of text, to move text on the screen, and to load and move various sorts of pictures was timed. While the extremes of these times ranged from 2 sec to 425 sec, a large number clustered around 6 sec/wait and that number was selected for use. Both of these numbers are entered on Form C (Figure 5.4).

5.23 Analysis of Ecology of Data

Step 7. Estimate frequency of operations

The different activities which might engage the user require different times to perform. The average time per page required by the user depends on the relative frequency with which he is required to perform these activities.

For the purposes of this problem a small sample was taken of articles appearing in *Cognitive Psychology*. For each article, the number of pages was counted as well as the number of headings, figures, footnotes, and paragraphs. The number of text segments requiring a special font treatment, such as italicized words, were counted (excluding those text segments already counted elsewhere, such as headings). Finally, the number of references was counted. These statistics are entered onto Form D (Figure 5.5). Each statistic is divided by the number of pages in the article and the result averaged over all the articles.

For many analysis purposes, these numbers would be sufficient. It is the claim of some industrial engineers who do time and motion studies on office work that sufficient accuracy is obtained by examining a single example of the work to be done as long as the example is not untypical (Birn, Crossan, and Eastwood, 1961; p. 288). For more precise results, one could examine a larger sample.

5.24 Compute Results

Step 8. List the operations needed for the problem.

At this point, the subsidiary calculations and data gathering have been completed and the final calculation can be made. The formating of a page in System A will require these operations:

FORM D
Frequency of Operations

Sample No.	No. Pages	Items Counted										
		Head.	Fig.	F.note	Para.	Font.	Refs.					
RAW COUNT												
1	31	21	9	4	72	39	94					
2	32	60	11	6	92	189	43					
3	22	30	5	4	84	94	24					
4	12	7	5	3	45	90	3					
5												
6												
7												
8												
9												
10												
11												
12												
13												
ITEMS/PAGE												
1	31	0.68	0.29	0.13	2.32	1.26	3.03					
2	32	1.88	0.34	0.18	2.62	5.91	1.34					
3	22	1.36	0.22	0.18	3.82	4.27	1.09					
4	12	0.58	0.41	0.25	3.75	7.50	0.25					
5												
6												
7												
8												
9												
10												
11												
12												
13												
AVERAGE ITMES/PAGE												
	24.25	1.12	0.24	0.15	3.13	4.42	1.43					

Figure 5.5. Frequency count for four journal articles

#Pp Process new page
#Ph Process headings
#Pf Process figure
#Pn Process footnotes
Hb Indent paragraph
Ft Specify text word font
#Pr Process refs

Five of these are macro operations, two are single unit tasks. Each of these operations is entered onto Form E (Figure 5.6).

Information needed in the calculation is transferred to Form E: from Form B, the number of unit tasks n_{uq} and waits n_{xq} for each of these operations is entered; from Form C, the time/unit task t_u and time/wait t_x are obtained.

Step 9. Compute the work time and wait time for each operation.

The time required by a user to perform operation q can be divided into working time w_q and time x_q spent waiting for the response of the machine. Working time can be computed from

$$w_q = n_{uq}t_u$$

The waiting time can be computed from

$$x_q = n_{xq}t_x.$$

Step 10. Compute work time/page, wait time/page, and total time/page for each operator.

By multiplying the working and waiting time for each operator by the expected number of occurrences of that operator per page $f_{q/p}$, we can obtain the expected working time per page $w_{q/p}$ and waiting time per page $x_{q/p}$.

$$w_{q/p} = f_{q/p}w_q$$

$$x_{q/p} = f_{q/p}x_q$$

The average time per page expected to be spent doing operator q is then the sum

$$T_{q/p} = w_{q/p} + x_{q/p}.$$

FORM E
Computation Summary

System: A

TASK ANALYSIS		SYSTEM ANALYSIS				ECOL.	RESULTS			
		NUMBER		TIME			TIME			
CODE	DESCRIPTION	UTs	Waits	Work	Wait	FREQ	Work	Wait	Total	%
		n_{uq}	n_{xq}	w_q (sec)	x_q (sec)	$f_{q/p}$	$w_{q/p}$ (sec)	$x_{q/p}$ (sec)	$T_{q/p}$ (sec)	
# Pp	Process new page	2	1	17.0	6.0	1.00	17.0	6.0	23.0	12%
# Ph	Process headings	4	0	34.0	0	1.12	38.1	0	38.1	20%
# Pf	Process figure	9	6	76.5	36.0	0.24	18.4	8.6	27.0	14%
# Pn	Process footnote	6	3	51.0	18.0	0.15	7.6	2.7	10.3	6%
Hb	Indent Paragraph	1	0	8.5	0	3.13	26.6	0	26.6	14%
Ft	Specify word font	1	0	8.5	0	4.42	37.6	0	37.6	20%
# Pr	Process refs	2	0	17.0	0	1.43	24.3	0	24.3	13%

$w_q = n_{uq}\ t_u$
$x_q = n_{xq}\ t_x$
$w_{q/p} = f_{q/p}\ w_q$
$x_{q/p} = f_{q/p}\ x_q$

Time/Ut	t_u	8.5	Totals	169.6	17.3	186.9	100%
Time/Wait	t_x	6.0	Add 25% error time			46.7	
Comp. Fact.	C	.8	Total with error time			233.6	

Figure 5.6. Computation of user time/page for System A

Step 11. Compute the total time/page.

The total time per page is just the sum of the operations:

$$T_p = \Sigma_q T_{q/p}$$

Computation of this number for the System A gives T_p = 186.9 sec. The wait time is 17.3 sec/page or about 9% of the time. The most time consuming operations are expected to be processing the headings and specifying text word fonts, each taking 20% of the time.

Step 13. Adjust for estimated error time.

The estimate so far has not allowed for the effect of errors. Since better data are not available, we shall use the 25% error time figure from Chapter 4. The total time/page, adjusted for error, is

$$T_p' = T_p + .25T_p$$

Adding in error time raises the estimated time/page for the problem to 233.6 sec/page.

Comparison with System B.

In System B, pages are numbered automatically, eliminating the page numbering step from *#Pp*. Headings are processed by specifying to which class they belong. The system then locates them, renders them in the correct font, and numbers them based on a previously specified style sheet. The work to be done for *#Ph* is reduced to a single unit task: *Th*, Specify heading type. System B also keeps track of the figure number so that captions are numbered automatically. The macro operations which are affected and the computation for System B are given in Figure 5.7 and Figure 5.8.

The increased automaticity of System B saves the user 83 sec/page. System B is predicted to require 152 sec/page, about 35% less time than System A. In System B, the wait time is the same 17.3 sec, but this is now 14% of the (error-free) time. Font changes to words in the text now account for 31% of the time.

FORM B
Macro Operations

Page 1 of 2

System: **B**

	ID	Description	Waits	UTs
MACRO OP	# Pp	Process new page	1	1
COMPONENT UNIT TASKS	Lp	Load Page	X	
MACRO OP	# Ph	Process heading	0	1
COMPONENT UNIT TASKS	Th	Specify heading type		
MACRO OP	# Pf	Process figure and caption	6	8
COMPONENT UNIT TASKS	Lf	Load Page	X	
	Vt	Specify vertical location of figure	X	
	Hf	Specify horizontal location of figure	X	
	Lc	Load caption	X	
	Vc	Specify vertical location of caption	X	
	Hc	Specify horizontal location of caption	X	
	Fc	Specify font of caption		
	Nt	Number figure callout in text		

Figure 5.7. Macro operations for System B (first page)

FORM E
Computation Summary

System: B

TASK ANALYSIS		SYSTEM ANALYSIS				ECOL.	RESULTS			
		NUMBER		TIME			TIME			
CODE	DESCRIPTION	UTs	Waits	Work	Wait	FREQ	Work	Wait	Total	%
		n_{uq}	n_{xq}	w_q (sec)	x_q (sec)	$f_{q/p}$	$w_{q/p}$ (sec)	$x_{q/p}$ (sec)	$T_{q/p}$ (sec)	
# Pp	Process new page	1	1	8.5	6.0	1.00	8.50	6.00	14.50	12%
# Ph	Process headings	1	0	8.5	0	1.12	9.52	0	9.52	8%
# Pf	Process figure	8	6	68.0	36.0	0.24	16.32	8.64	24.96	21%
# Pn	Process footnotes	6	3	51.0	18.0	0.15	7.65	2.70	10.35	9%
# Ft	Process text fonts	1	0	8.5	0	4.42	37.57	0	37.57	31%
# Pr	Process refs	2	0	17.0	0	1.43	24.31	0	24.31	20%

$w_q = n_{uq}\ t_u$
$x_q = n_{xq}\ t_x$
$w_{q/p} = f_{q/p}\ w_q$
$x_{q/p} = f_{q/p}\ x_q$

Time/Ut	t_u	8.5	Totals	103.87	17.34	121.21
Time/Wait	t_x	6.0	Add 25% error time			30.30
Comp. Fact.	C	.8	Total with error time			151.51

Figure 5.8. Computation of user time/page for System B

5.3 INTERACTIONS AMONG UNIT TASKS

The above computations assumed that the time to perform a unit task is independent of the surrounding unit tasks. In fact, unit tasks are not independent of each other, but the assumption is quite reasonable for this analysis.

The macro operation *#Ph, Process heading,* will, according to Figure 5.3 be composed of four unit tasks:

Fh *Specify font of heading*
Vh *Specify vertical spacing of heading*
Hh *specify horizontal placement of heading*
Nh *Specify heading number.*

Expanding these to GLMWV steps gives

(*Fh*	*Specify font of heading*)	
	Detect heading	G
	Point to entire heading	L
	Change font of heading	M
	Verify	V
(*Vh*	*Specify vertical spacing of heading*)	
	Detect heading	G
	Point to heading	L
	Insert lines before heading	M
	Insert lines after heading	M
	Verify	V
(*Hh*	*Specify horizontal placement of heading*)	
	Detect heading	G
	Point to heading	L
	Move heading	M
	Verify	V
(*Nh*	*Specify heading number*)	
	Detect heading	G
	Point to heading number	L
	Insert number	M
	Verify	V.

But if the user performs these unit tasks one after another he need not do all of the suboperations. For example, once he has selected the heading, in unit task *Fh,* he need not do it again in *Vh* and *Hh.* Rewriting with the redundant steps eliminated gives

(*Fh*	*Specify font of heading*)	
	Detect heading	**G**
	Point to entire heading	**L**
(*Vh*	*Specify vertical spacing of heading*)	
	Detect heading	**G**
	Point to heading	**L**
	Insert lines before heading	**M**
	Insert lines after heading	**M**
	Verify	**V**
(*Hh*	*Specify horizontal placement of heading*)	
	Move heading	**M**
(*Nh*	*Specify heading number*)	
	Point to heading number	**L**
	Insert number	**M**
	Verify	**V.**

Instead of the 16 GLMV suboperations that it nominally takes to do the four unit tasks separately, it only takes 9 suboperations to do the macro operation *#Ph*. This gives a compression factor of 9/16 = .56. Notice, however, that while the detailed examination showed that some suboperations are redundant and may be eliminated, the analysis also revealed that, as in the case of *Vh*, some suboperations have extra elements (here, the two M suboperations). Repeating this analysis for each macro operation shows that the number of extra operations revealed is nearly equal to the number of redundant operations and the two cancel. The assumption of independence between unit tasks is a good approximation for the analysis.

To show this, the analysis is carried through, recording the number of GLMWV operations for each macro unit task in Figure 5.9. The individual compression factors (Since **WAITS** are dealt with separately, they do not enter the calculation for the compression factor) for each of the macro operations are

#Pp	*Process new page*	1.00
#Ph	*Process heading*	0.56
#Pf	*Process figure*	0.97
#Pn	*Process footnote*	1.17
#Pr	*Process reference*	1.00
	Average	0.94

Since the average compression ratio is within a few percent of 1.0 (that is, no compression) it can be ignored.

FORM B2
Macro Operations

Page 1 of 2

System: A

	ID	Description	G	L	M	W	V	UTs
MACRO OP	# Pp	Process new page setup	3	1	3	1	1	2
COMPONENT UNIT TASKS	Lp	Load page	1		1	1		
	Np	Specify page number	2	1	2		1	
MACRO OP	# Ph	Process heading	1	2	5	0	1	4
COMPONENT UNIT TASKS	Fh	Specify font of heading	1	1	1			
	Vh	Specify vertical spacing of heading			2			
	Hh	Specify horizontal placement of heading			1			
	Nh	Specify heading number		1	1		1	
MACRO OP	# Pf	Process figure and caption	8	10	9	6	8	9
COMPONENT UNIT TASKS	Lf	Load figure	1	1	1	1	1	
	Vf	Specify vertical location of figure	1	1	1	1	1	
	Hf	Specify horiztonal location of figure	1	1	1	1	1	
	Lc	Load caption	1	1	1	1	1	
	Vc	Specify vertical location of caption	1	1	1	1	1	
	Hc	Specify horizontal location of caption	1	1	1	1	1	
	Fc	Specify font of caption		2	1			
	Nc	Specify number of caption	1	1	1		1	
	Nt	Number figure callout in text	1	1	1		1	

Figure 5.9. Re-analysis of macro operations for System A by number of GLMWV suboperations

FORM B2
Macro Operations

Page 2 of 2

System: A

	ID	Description	G	L	M	W	V	UTs
MACRO OP	# Pn	Process footnote	7	9	6	3	6	6
COMPONENT UNIT TASKS	Ln	Load footnote	1	1	1	1	1	
	Vn	Specify vertical location of footnote	1	2	1	1	1	
	Hn	Specify horizontal location of footnote	1	2	1	1	1	
	Fn	Specify footnote font	1	2	1		1	
	Nn	Number footnote body	2	1	1		1	
	Nt	Number footnote callout	1	1	1		1	

MACRO OP	# Pr	Process reference	2	2	2	0	2	2
COMPONENT UNIT TASKS	Ft	Italicize journal name	1	1	1		1	
	Ft	Render volume number in boldface	1	1	1		1	

MACRO OP								
COMPONENT UNIT TASKS								

Figure 5.9. (continued)

5.4 GLMWV METHOD

Having gone to the effort to compute the number of G, L, M, W, and V suboperations in each macro operation, it is possible to compute from measured times for these suboperations the answer to the problem. Reasonable values for the required time to perform each suboperation can be read from Table 4.2. These values (rounded to the nearest .5 sec to emphasize their approximate nature) are G: 2.5 sec, L(command): 4.0 sec, L(pointing): 2.0 sec, M: 2.5 sec, and V: 1.0 sec. W was previously estimated to be 6.0 sec. Figure 5.10 gives the computation. This computation is a refinement of the computation using only the number of unit tasks and the number of waits which appeared in Figure 5.6. The resulting (error-free) time per page from the new calculation is 200.9 sec as compared with 186.9 sec computed in Figure 5.6, a difference of only about 7%. Thus, for a conceptual design problem in which rough figures will suffice, the unit task calculation method gives a close enough approximation to the GLMWV method and with less effort.

5.5 CAVEATS

With a simple model of this sort caveats must be made. The most important caveat is the extremely superficial analysis of the task environment. The method is not suitable for detecting significant interactions or of discovering that some other phenomenon (e.g., memory load) plays a dominant role. Thus it is possible that the analysis is simply wide of the mark in some gross way.

Further consideration reveals several sources of variability not considered in the above calculations. There is variability of factors of about two depending on the skill of the operator, and of (independent) factors of about two depending on the structure of the command and display system (e.g., RCG vs Poet editors) and of some (unknown) factor depending on the difficulty of the actual manuscript material. A final source of the variability comes directly from the crudity of the method analysis. The actual methods are substantially more conditional than the simple linear analyses used.

The current analysis does not deal with errors directly. Since errors do get converted into extra time, one can develop a factor to estimate the "expansion of time due to error" and this is what we have done. An important issue on errors is the impact of big errors—of the ones that occur with relatively low frequency, but take 10, even 20, times the mean unit task time for recovery. It is possible that such errors could be a significant fraction of the total time.

FORM E 2
Computation Summary System: A

TASK ANALYSIS		SYSTEM ANALYSIS										ECOL	RESULTS						
		NUMBER					TIME						TIME						
CODE	DESCR	G	L	M	W	V	t_G	t_L	t_M	t_W	t_V	FREQ $f_{q/p}$	T_G	T_L	T_M	T_W	T_V	TOT	%
# Pp	New page	3	1	3	1	1	7.5	4.0	7.5	6.0	1.0	1.00	7.5	4.0	7.5	6.0	1.0	26.0	13%
# Ph	Headings	1	2	5	0	1	2.5	8.0	12.5	0	1.0	1.12	2.8	9.0	14.0	0	1.1	26.9	13%
# Pf	Figures	8	10	9	6	8	20.0	40.0	22.5	36.0	8.0	0.24	4.8	9.6	5.4	8.6	1.9	30.3	15%
# Pn	Footnotes	7	9	6	3	6	17.5	36.0	15.0	18.0	6.0	0.15	2.6	5.4	2.2	2.7	0.9	13.8	7%
Hb	Indent para.	1	1	1	0	1	2.5	4.0	2.5	0	1.0	3.13	7.8	12.5	7.8	0	3.1	31.2	16%
Ft	Text fonts	1	1	1	0	1	2.5	4.0	2.5	0	1.0	4.42	11.0	17.7	11.0	0	4.4	44.1	22%
# Pr	References	2	2	2	0	2	5.0	8.0	5.0	0	2.0	1.43	7.2	11.4	7.1	0	2.9	28.6	14%

Time/GET	t_G	2.5	Totals	43.7	69.6	55.0	17.3	15.3	200.9
Time/LOCATE	t_L	4.0	Add 25% error time						50.2
Time/MODIFY	t_M	2.5	Total with error time						251.1
Time/WAIT	t_W	6.0							
Time/VERIFY	t_V	1.0							

Figure 5.10. Recomputation of user time/page for System A using times for GLMWV suboperations

The method described in this chapter is most suitable for use at the conceptual design level. What the method provides is a quick and simple method for obtaining order of magnitude estimates of the time required by the user to perform a task with a given sort of machine. User's time can, of course, be converted into costs and used in an economic analysis.

6

Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT†

An important element in the design of the man-computer interface is the method of pointing by which the user indicates to the computer his selection of some element on the computer display. This is especially important for text-editing programs using a CRT display (such as RCG and Editor Y) where the user may repeatedly use a pointing device to select the text he wishes to modify or to invoke a command from a menu

† Adapted from Card, English, and Burr (in Press).

displayed on the screen. The choice of pointing device may have a significant impact on the ease with which the selections can be made, and hence, since pointing typically occurs with high frequency, on the success of the entire system. In this chapter, we shall focus in on the psychology of this important component of a text editing system.

English, Englebart, and Berman (1967) measured mean pointing times and error rates for the mouse, lightpen, Grafacon tablet, and position and rate joysticks. They found the mouse to be the fastest of the devices, but did not investigate the effect of distance to target. They also gave no indication of the variability of their measures. Goodwin (1975) measured pointing times for the lightpen, lightgun, and Saunders 720 step keys. She found the light pen and lightgun equally fast and much superior to the Saunders 720 step keys. However, she used only one target size and did not investigate distance. In addition, her results also show large learning effects which are confounded with the device comparisons. Both studies were more concerned with the evaluation of devices than with the development of models from which performance could be predicted. In another line of development Fitts and others (Fitts, 1954; Fitts and Peterson, 1964; Fitts and Radford, 1966; Knight and Dagnal, 1967; Welford, 1968) developed and tested the relation between distance, size of target, and hand movement time. Such a relation might potentially be used to predict pointing times for devices involving continuous hand movements; however this has not been tested directly. In particular it was not known whether Fitts's Law would hold for targets of the shape and character of text strings.

This chapter examines text selection performance with four devices: the mouse, a rate-controlled isometric joystick, step keys, and text keys. The study differs from the English et al. and Goodwin studies in that distance, target size, and learning are all simultaneously controlled and a different set of devices is measured. Also, unlike those studies, an attempt is made to give a theoretical account of the results. In particular, performance on the continuous movement devices is tested against the predictions of Fitts's Law.

6.1 METHOD

Subjects

Three men and two women, all undergraduates at Stanford University, served as subjects in the experiment. None had ever used any of the devices previously and all had little or no experience with computers. Subjects were paid $3.00 per hour with a $20.00 bonus for

completing the experiments. One of the five subjects was very much slower than the others and was eliminated from the experiment.

Pointing Devices

Four pointing devices were tested (see Plate 6.1). Two were continuous devices: the mouse and a rate-controlled isometric joystick. Two were key operated: the step keys and the text keys. The devices had been optimized informally by testing them on local users, adjusting the device parameters so as to maximize performance.

The mouse, a version of the device described in English et al. (1967), was a small device which sat on the table to the right of the keyboard, connected by a thin wire. On the undercarriage were two small wheels, mounted at right angles to each other. As the mouse moved over the table one wheel coded the amount of movement in the X-direction, the other the movement in the Y-direction. As the mouse moved, a cursor moved simultaneously on the CRT, two units of screen movement for each unit of mouse movement.

The joystick used was a small strain gauge on which had been mounted a rubber knob 1.25 cm in diameter. Applying force to the joystick in any direction did not produce noticeable movement in the joystick itself, but caused the cursor to move in the appropriate direction at a rate = 0.0178 $(\text{force})^2$ in cm/sec, where force is measured in Newtons. For forces less than about 4 Newtons, the cursor did not move at all, and the equation ceased to hold in the neighborhood of 45 Newtons as the rate approached a ceiling of about 40 cm/sec.

The step keys were the familiar five key cluster found on many CRT terminals. Surrounding a central HOME key were keys to move the cursor in each of four directions. Pressing the HOME key caused the cursor to go to the upper left corner of the text. Pressing one of the horizontal keys moved the cursor 1 character (0.246 cm on the average) along the line. Pressing a vertical key moved the cursor one line (0.456 cm) up or down. Holding down one of the keys for more than 0.100 sec caused it to go into a repeating mode, producing one step in the vertical direction each 0.133 sec or one step in the horizontal direction each 0.067 sec (3.43 cm/sec vertical movement, 3.67 cm/sec horizontal movement).

The text keys were similar to keys appearing on several commercial "word processing" terminals. Depressing the PARAGRAPH key caused the cursor to move to the beginning of the next paragraph. Depressing the LINE key caused the cursor to move downward to the same position in the next line. The WORD key moved the cursor forward one word; the CHARACTER key moved the cursor forward one character. Holding down the REVERSE key while pressing another key caused the

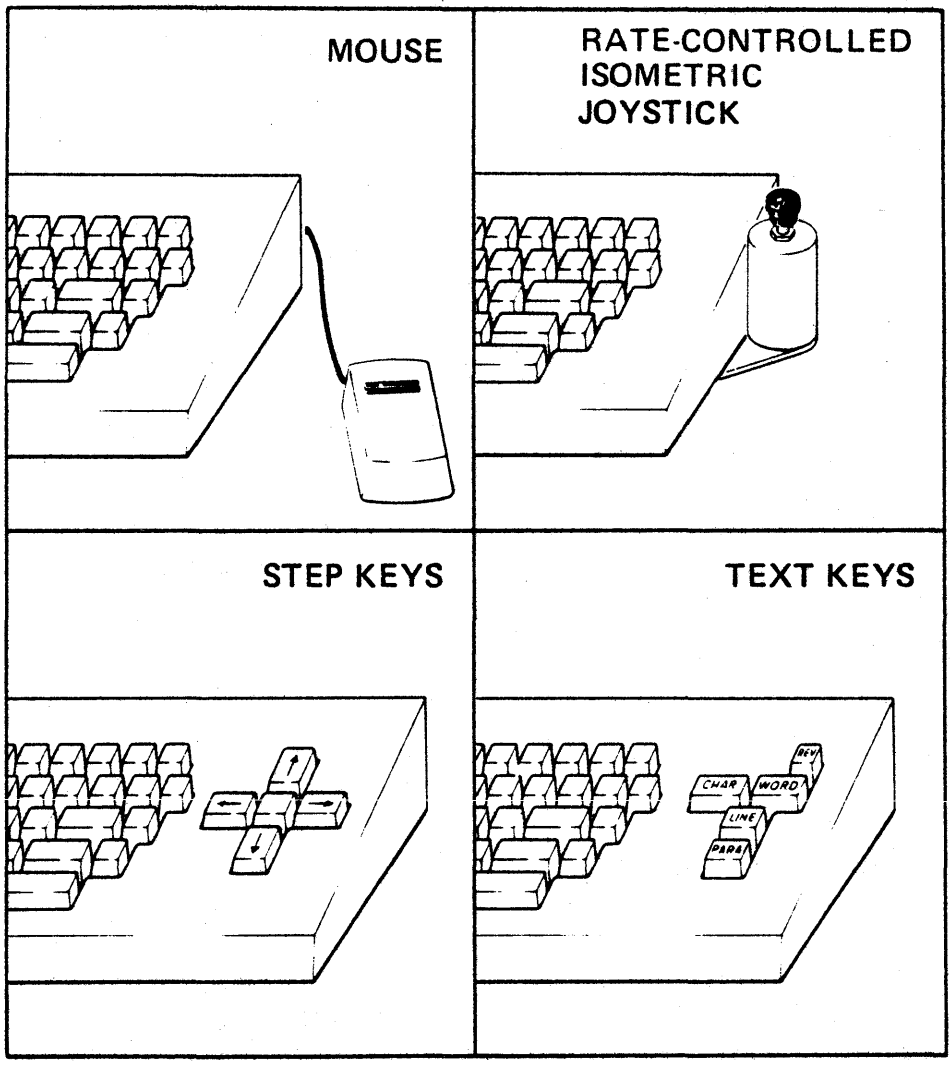


Plate 6.1. Pointing devices tested.

cursor to move opposite the direction it would otherwise have moved. The text keys could also be used in a repeating mode. Holding the LINE, WORD, or CHARACTER keys down for longer than 0.100 sec caused that key to repeat at 0.133 sec per repeat for the LINE key, 0.100 sec per repeat for the WORD key, or 0.067 sec per repeat for the CHARACTER key. Since there were 0.456 cm/line, 1.320 cm/word, and 0.246 cm/character movement rates were 3.43 cm/sec for the LINE key, 13.20 cm/sec for the WORD key, and 3.67 cm/sec for the CHARACTER key.

Procedure

Subjects were seated in front of a computer terminal with a CRT for output, a keyboard for input, and one of the devices for pointing at targets on the screen. On each trial a page of text was displayed on the screen. Within the text a single word or phrase, the target, was highlighted by inverting the black/white values of the text and background in a rectangle surrounding the target. The subject struck the space bar of the keyboard with his right hand, then, with the same hand reached for the pointing device and directed the cursor to the target. The cursor thus positioned, the subject pressed a button "selecting" the target as he would were he using the device in a text editor. For the mouse, the button was located on the device itself. For the other devices, the subject pressed a special key with his left hand.

Design

Text selections and targets were so arranged that there were five different distances from starting position to target, 1, 2, 4, 8, or 16 cm, and four different target sizes, 1, 2, 4, or 10 characters. All targets were words or groups of words. Ten different instances of each distance × target size pair were created, varying the location of the target on the display and the angle of hand movement to give a total of 200, randomly ordered, unique stimuli.

Each subject repeated the experiment with each device. The order in which subjects used the devices was randomized. At the start of each day, the subjects were given approximately twenty warmup trials to refresh their memory of the procedure. All other trials were recorded as data. At the end of each block of twenty trials they were given feedback on the average positioning time and average number of errors for those trials. This feedback was found to be important in maintaining subjects' motivations. At the end of each 200 trials they were given a rest break of about fifteen minutes. Subjects normally accomplished 600 trials/day involving about two to three hours of work. They each used a particular

device until the positioning time was no longer decreasing significantly with practice (operationally defined as when the first and last thirds of a block of the last 600 trials excluding the first 200 trials of a day did not differ significantly in positioning time at the $p < 0.05$ level using a *t*-test). An approximation to this criteria was reached in from 1200 to 1800 trials (four to six hours) on each device. Of the 20 subject × device pairs, 15 reached this criterion, 3 performed worse in their last trials (largely because some time elapsed between sessions), and only 2 were continuing (slightly) to improve.

6.2 RESULTS

Improvement of Performance with Practice

The learning curve which gives positioning time as a function of the amount of practice can be approximated (De Jong, 1957) by

$$T_N = T_1 N^{-\alpha} \qquad (6.1)$$

where

T_1 = estimated positioning time on the first block of trials,
T_N = estimated positioning time on the Nth block of trials,
N = trial block number, and
α = an empirically determined constant.

This form is convenient since taking the log of both sides produces an equation linear in log N,

$$\log T_N = \log T_1 - \alpha (\log N). \qquad (6.2)$$

Thus the ease of learning for each device can be described by two numbers T_1 and α, which numbers may be conveniently determined empirically by regressing log T_N on log N. Plate 6.2 shows the results of plotting the data from error-free trials according to Equation 6.2. Each point on the graph is the average of a block N of twenty contiguous trials from which error trials have been excluded. Only the first 60 trial blocks are shown. Since some subjects reached criterion at this point, not all continued on to further trials. The values predicted by the equation are given as the straight line drawn through the points. The average target size in each block was 4.23 cm (the range of the average target sizes for different trial blocks was 3.95 to 4.50 cm); the average distance

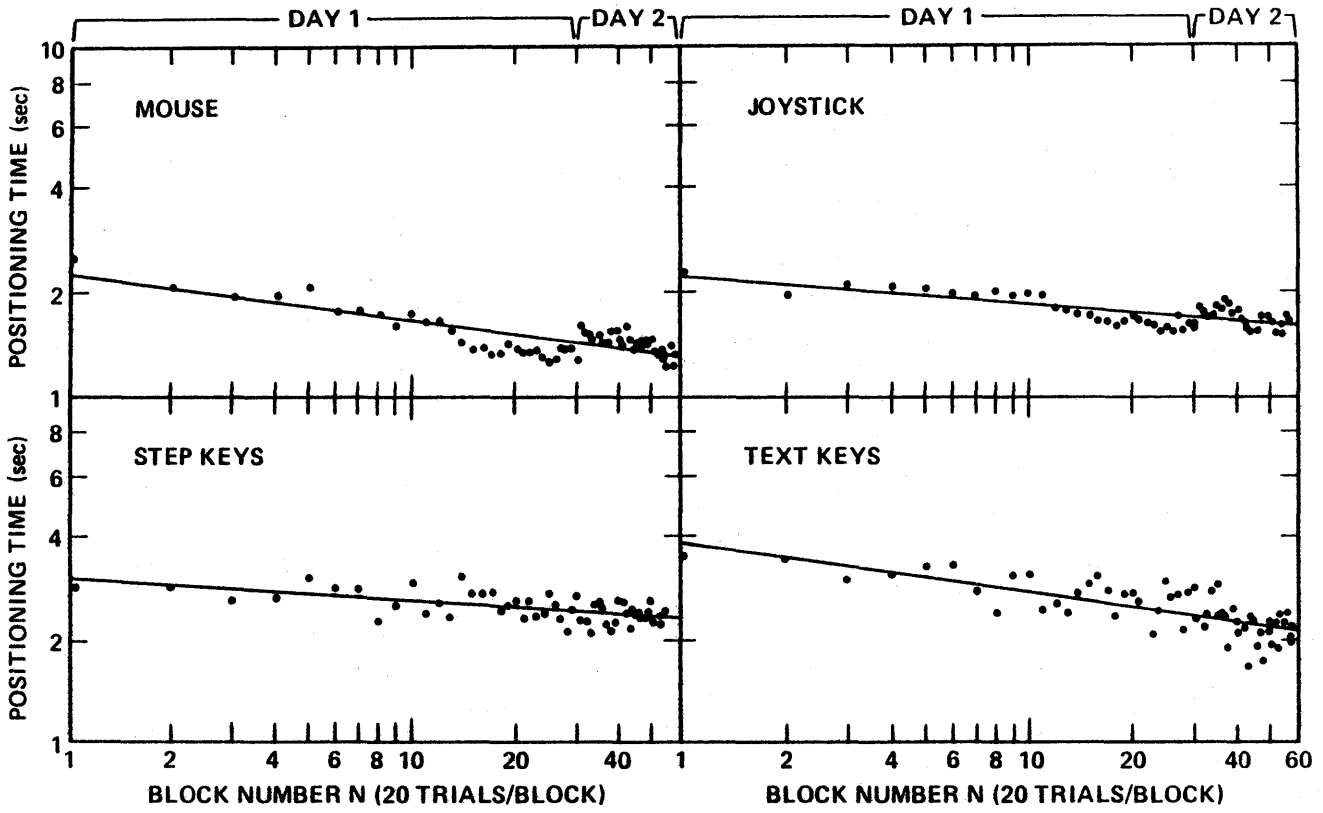


Plate 6.2. Learning curves for pointing devices.

to the target was 6.13 cm (range 5.90 to 6.42 cm).

The parameters T_1 and α, as determined by the regressions, are given in Table 6.1, along with the standard error and squared multiple correlation from the regression analysis. Practice causes more improvement in the mouse and text keys than on the other two devices. The step keys, in particular, show very little improvement with practice. Equation 6.2 explains 39% of the variance in the average positioning time for a block of trials for the step keys, 61% to 66% of the variance for the other devices. The fit, at least for the mouse and the joystick, is actually better than these numbers suggest. Since subjects did 30 blocks of trials on a day typically followed by a pause of a day or two before they could be rescheduled, a break in the learning curve is expected at that point and indeed such a break is quite evident for the mouse and the joystick between the 30th and 31st blocks. Fitting Equation 6.2 to only the first day increases the percentage of variance explained to 91% for the mouse and 83% for the joystick. In the case of the step keys and text keys there is no such obvious day effect.

Overall Speed

In order to compare the devices after learning has nearly reached asymptote (as would be the case for office workers using them daily), a sample of each subject's performance on each device was examined consisting of the last 600 trials excluding the first 200 trials of a day (in order to diminish warmup effects). The remaining analyses will be based on this subset of the data, excluding those trials on which errors occurred. Table 6.2 gives the homing time, positioning time, and total time for each device averaging over all the distances and target sizes. *Homing time* was measured from the time the subject's right hand left the space bar until the cursor had begun to move. *Positioning time* was measured from when the cursor began to move until the selection button had been pressed. From the table, it can be seen that homing time increases slightly with the distance of the device from the keyboard. The longest time required is to reach the mouse, the shortest to reach the step keys. Although the text keys are near the keyboard, they take almost as long to reach as the mouse. Either it is more difficult to position the hands on the text keys or, as seems likely, subjects often spent some time planning the strategy for their move in the time between hitting the space bar to start the clock and the time when they begin pressing the keys. Further evidence for this hypothesis comes from the relatively high standard deviation observed for the homing time of the text keys. While the differences in the homing times among all device pairs except the mouse vs. the text keys are reliable statistically (at $p < 0.05$ or better

TABLE 6.1

LEARNING CURVE PARAMETERS

DEVICE	T_1 (sec)	α	Learning Curve Equation[a]	s_e (sec)	R^2
Mouse	2.20	0.13	$T_N = 2.20\ N^{-0.13}$	0.12	0.66
Joystick	2.19	0.08	$T_N = 2.19\ N^{-0.08}$	0.08	0.62
Step Keys	3.03	0.07	$T_N = 3.03\ N^{-0.07}$	0.11	0.39
Text Keys	3.86	0.15	$T_N = 3.86\ N^{-0.15}$	0.16	0.61

[a] N is number of trial blocks. There are 20 trials in each block.

TABLE 6.2

OVERALL TIMES

DEVICE	Movement time for non-error trials (sec)						Error Rate	
	Homing Time		Positioning Time		Total Time			
	M	SD	M	SD	M	SD	M	SD
Mouse	0.36	0.13	1.29	0.42	1.66	0.48	5%	22%
Joystick	0.26	0.11	1.57	0.54	1.83	0.57	11%	31%
Step Keys	0.21	0.30	2.31	1.52	2.51	1.64	13%	33%
Text Keys	0.32	0.61	1.95	1.30	2.26	1.70	9%	28%

using a *t*-test), the differences are actually quite small. For example, while the step keys can be reached 0.15 sec sooner than the mouse, they take 1.02 sec longer to position. Thus the differences in the homing times are insignificant compared to the differences between the positioning times.

The mouse is easily the fastest device, the step keys the slowest. As a group, the continuous devices (the mouse and the joystick) are faster than the key-operated devices (the step keys and text keys). Differences between the devices are all reliable at $p \ll 0.001$ using *t*-tests.

Effect of Distance and Target Size

The effect of distance on positioning time is given in Plate 6.3. At all distances greater than 1 cm, the continuous devices are faster. The positioning time for both continuous devices seems to increase approximately with the log of the distance. The time for the step keys increases rapidly as the distance increases, while the time for the text keys increases somewhat less than as the log of the distance, owing to the existence of keys for moving relatively large distances with a single stroke. Again the mouse is the fastest device, and its advantage increases with distance.

Plate 6.4 shows the effect of target size on positioning time. The positioning time for both the mouse and the joystick decreases with the log of the target size. The time for the text keys is independent of target size and the positioning time for the step keys also decreases roughly with the log of the target size. Again the mouse is the fastest device, and again the continuous devices as a group are faster for all target sizes.

Effect of Approach Angle

The targets in text editing are rectangles often quite a bit wider than they are high. Hence they might present a different problem when approached from different angles. In addition, the step keys and text keys work somewhat differently when moving horizontally than when moving vertically. To test if the direction of approach has an effect on positioning time, the target movements were classified according to whether they were vertical (0 to 22.5 degrees) diagonal (22.5 degrees to 67.5 degrees) or horizontal (67.5 degrees to 90 degrees). Analysis of variance shows the angle makes a significant difference in every case except for the mouse. The joystick takes slightly longer to position when the target is approached diagonally. The step keys take longer when approached horizontally than when approached vertically, a consequence probably deriving from the fact that a single keystroke would move the

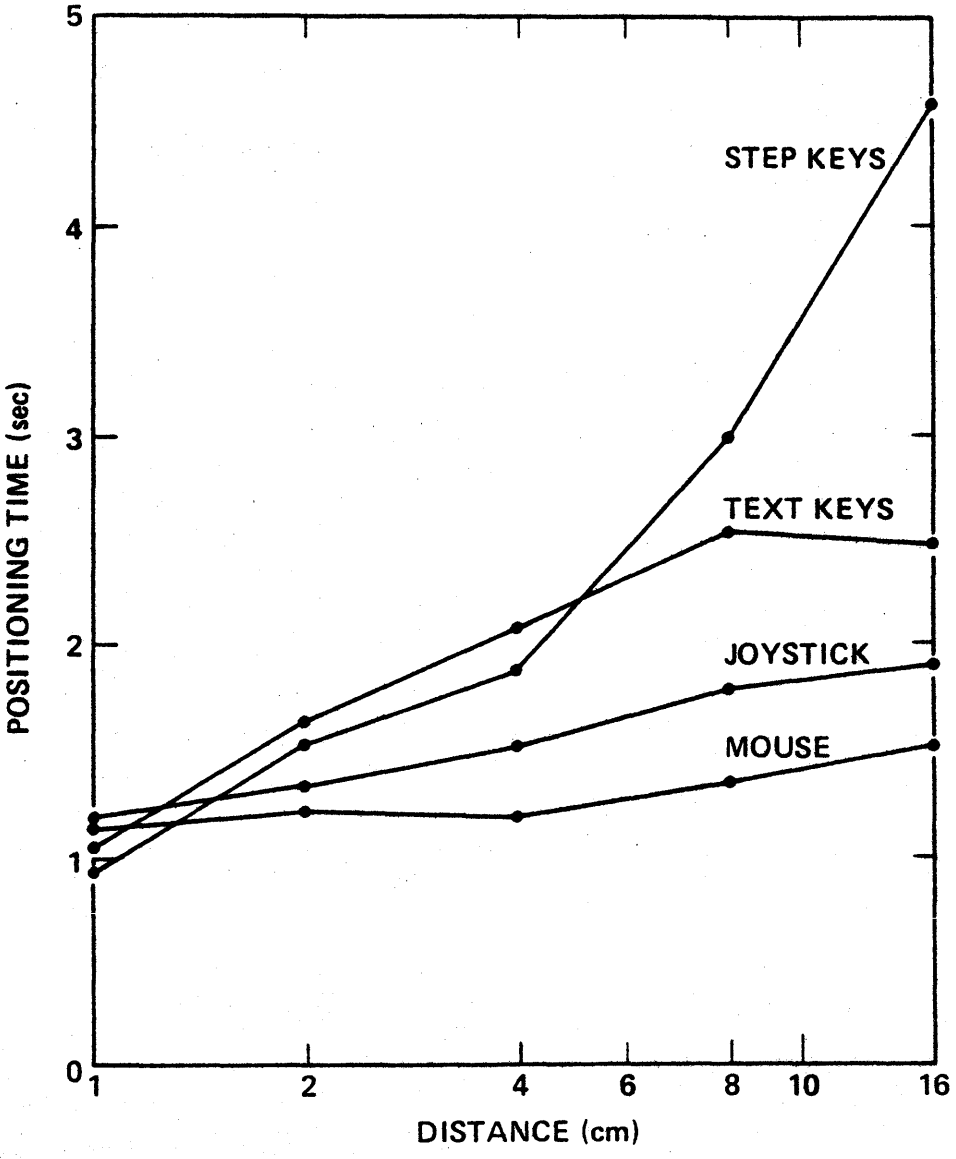


Plate 6.3. Effects of target distance on positioning time.

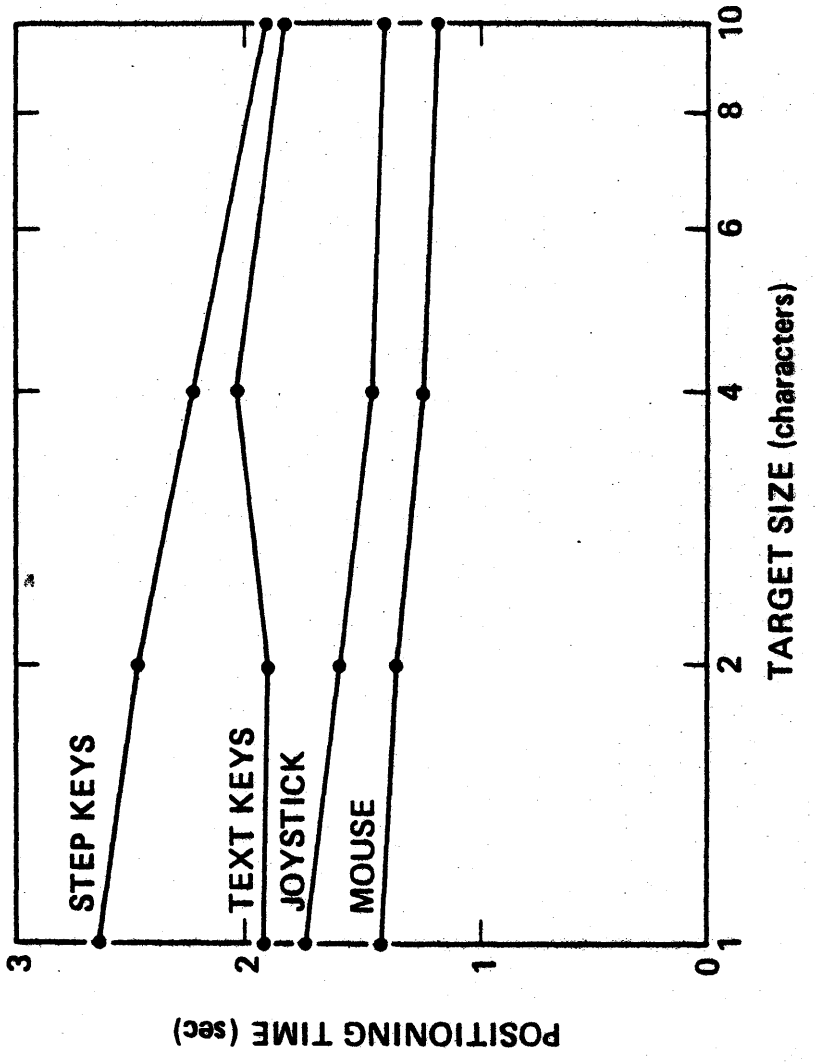


Plate 6.4. Effect of target size on positioning time.

cursor almost twice as far vertically as horizontally. By contrast, the text keys take longer to position vertically, reflecting the presence of the WORD key. The differences induced by direction are not of great consequence, however. For the joystick it amounts to 3% of the mean positioning time; for the step keys 9%; for the text keys 5%.

Errors

Of the four devices tested, the mouse had the lowest overall error rate, 5%; the step keys had the highest, 13%. The differences are reliable at $p < 0.05$ or better using t-tests. There is only a very slight increase in error rate with distance. However, there is a decrease in error rate with target size for every device except the text keys (Plate 6.5). This finding replicates the result of Fitts and Radford (1966). In an investigation of self-initiated, discrete, pointing movements using a stylus, there was a similar marked reduction in errors as the target increased in size, but only a slight increase in error rate as the distance to the target increased.

6.3 DISCUSSION

While these empirical results are of direct use in selecting a pointing device, it would obviously be of greater benefit if a theoretical account of the results could be made. For one thing, the need for some experiments might be obviated; for another, ways of improving pointing performance might be suggested. Fortunately, a first-order account for the devices of this experiment is not hard to give.

Mouse

The time to make a hand movement can be described by a version of Fitts's Law (Welford, 1968),

$$T_{pos} = K_0 + K \log_2 (D/S + 0.5) \text{ sec} \qquad (6.3)$$

where

T_{pos} = Positioning time,
D = Distance to the target,
S = Size of the target, and
K_0, K = constants.

Here the constant K_0 includes within it the time for the hand initially to adjust its grasp on the mouse and the time to make the selection with the

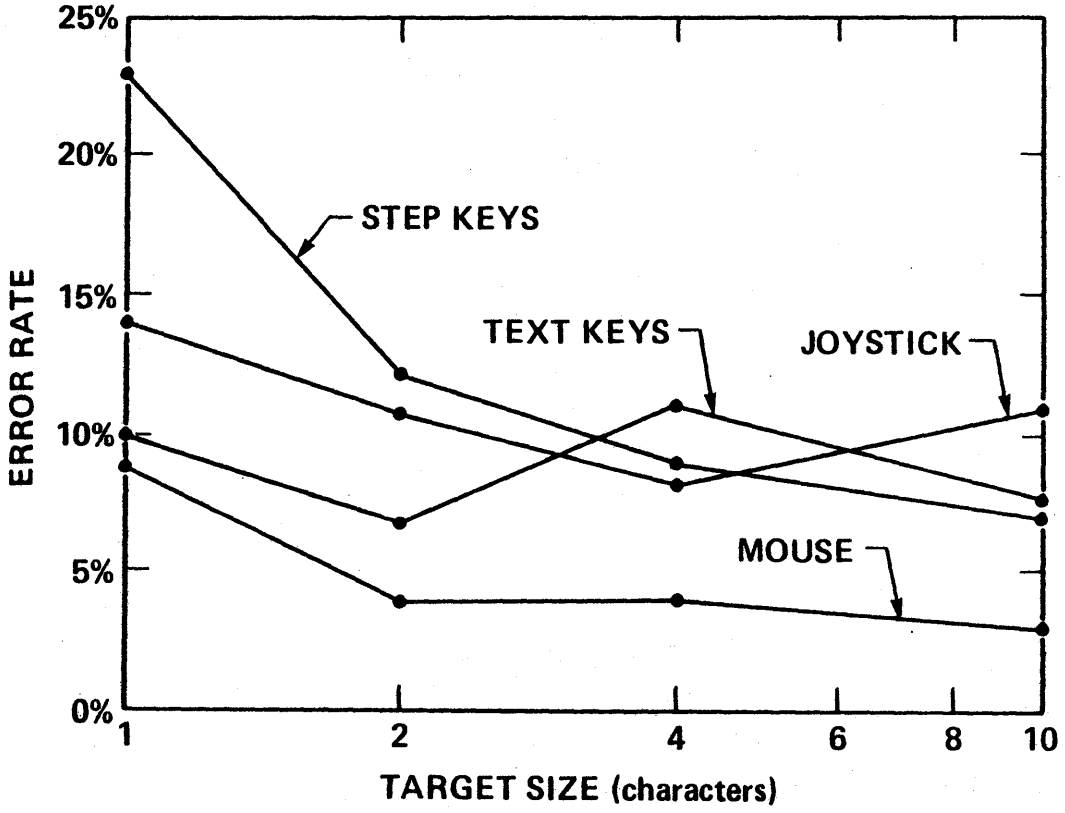


Plate 6.5. Effect of target size on error rate.

selection button. A constant of $K \simeq 0.1$ sec/bit (10 bits/sec) appears in a large number of studies on movement. This number is a measure of the information processing capacity of the eye-hand coordinate system. For single, discrete, subject-paced movements, the constant is a little less than 0.1 sec/bit. Fitts and Radford (1966) get a value of 0.078 sec/bit (12.8 bits/sec, recomputed from their Figure 1, Experiment I, for the experimental condition where accuracy is stressed). Pierce and Karlin (1957) get maximum rates of 0.085 sec/bit (11.7 bits/sec) in a pointing experiment. For continuous movement, repetitive, experimenter-paced tasks, such as alternately touching two targets with a stylus or pursuit tracking, the constant is slightly above 0.1 sec/bit. Elkind and Sprague (1961) get maximum rates of 0.135 sec/bit (7.4 bits/sec) for a pursuit tracking task. Fitts's original dotting experiment as replotted by Welford (1968, p. 148) gives a K of 0.120 sec/bit as does Welford's own study using the actual distance between the dots, the same measure of distance used in this study.

Fitts's Law predicts that plotting positioning time as a function of $\log_2(D/S + 0.5)$ should give a straight line. As the solid line in Plate 6.6 shows, this prediction is confirmed. Furthermore, the slope of the line K should be in the neighborhood of 0.1 sec/bit. Again the prediction is confirmed. The equation for the line in Plate 6.6 as determined by regression analysis is

$$T_{pos} = 1.03 + 0.096 \log_2 (D/S + 0.5) \text{ sec.} \qquad (6.4)$$

The equation has a standard error of 0.07 sec and explains 83% of the variance of the means for each condition. This is roughly comparable to the percentage of variance explained by Fitts and Radford. The slope of 0.096 sec/bit is in the 0.1 sec/bit range found in other studies. Since the standard error of estimate for K is 0.008 sec/bit, the mouse would seem to be close to, but slightly slower than, the optimal rate of around 0.08 sec/bit observed for the stylus and for finger pointing.

The values for positioning time obtained in this experiment are apparently in good agreement with those obtained by English et al. Making the assumption that their CRT characters were about the same width as ours and assuming an intermediate target distance of about 8 cm, Equation 6.4 (plus the addition of the 0.36 sec homing time from Table 6.2) predicts 1.87 sec for 1 character targets (English et al. got 1.93 sec) and 1.66 sec for "word" targets of 5 characters (they got 1.68 sec).

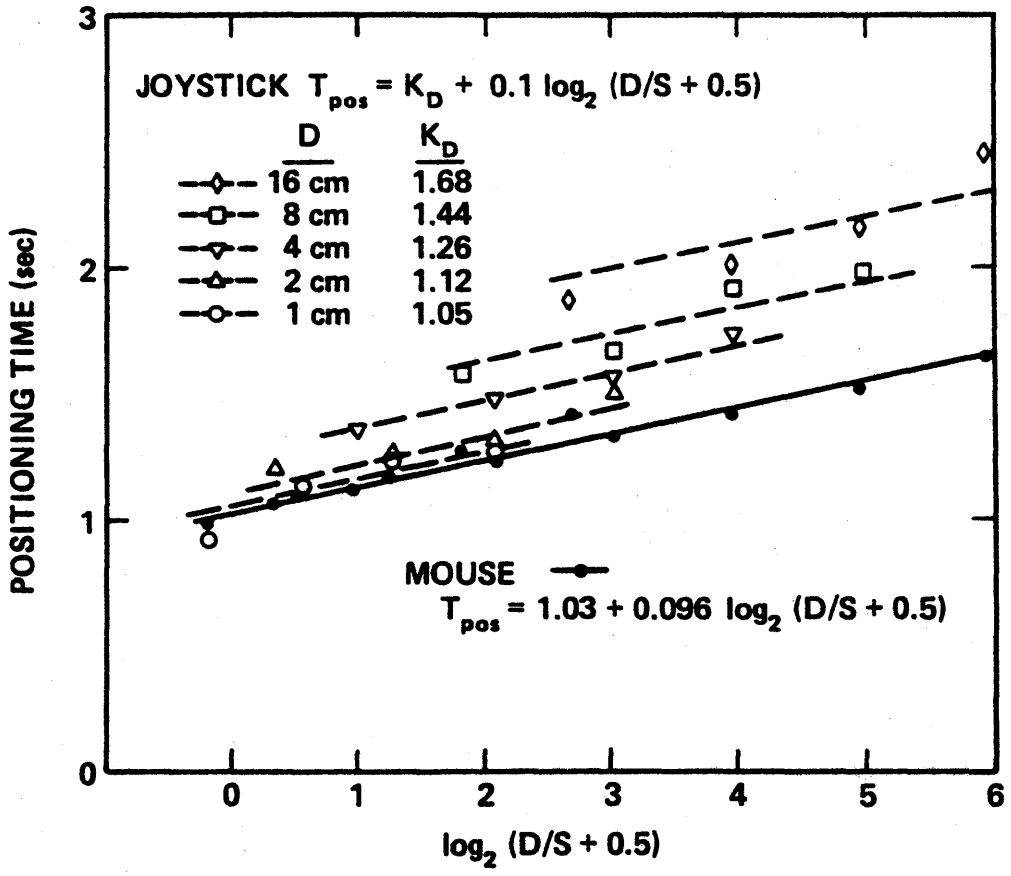


Plate 6.6. Positioning time for continuous devices as a function of Fitts's Index of Difficulty $\log_2 (D/S + 0.5)$

Joystick

Although it is a rate-controlled device instead of a position device, we might wonder if the joystick follows Fitts's Law. Plotting the average time per positioning for each distance × size cell of the experiment according to Equation 6.3 shows that there is an approximate fit to

$$T_{pos} = 0.99 + 0.220 \log_2 (D/S + 0.5). \quad (6.5)$$

Equation 6.5 has a standard error of 0.13 sec and explains 89% of the variance of the means. The size of the slope K shows that information is being processed at only half the speed as with the mouse and significantly below the maximum rate. Closer examination gives some insight into the difficulty. The points for the joystick in Plate 6.6 actually form a series of parallel lines, one for each distance, each with a slope of around 0.1 sec/bit. Setting K to 0.1 sec/bit, we can therefore write as an alternative model

$$T_{pos} = K_D + 0.1 \log_2 (D/S + 0.5).$$

K_D is the intercept for distance D. From the figure, K_D is about 1.05 sec for D = 1 cm, 1.12 sec for 2 cm, 1.26 sec for 4 cm, 1.44 sec for 8 cm, and 1.68 sec for 16 cm. For this model the standard error of the fit is reduced to 0.07 sec, the same as for the mouse. (Since the slope was not determined by the regression, a comparable R^2 cannot be computed.) Thus the tested joystick can be thought of as a Fitts's Law device with a slope twice that for hand movements; or it can be thought of as a Fitts's Law device with the expected slope, but having an intercept which increases with distance. The problem with this joystick is probably related to the non-linearity in the control (Poulton, 1974; Craik and Vince, 1963). It should be noted that for the 1 cm distance (where the effect of non-linearity is slight) the positioning time is virtually the same as for the mouse. Thus the possibility of designing a joystick with performance characteristics comparable to the mouse is by no means excluded.

Step Keys

As a first approximation one might expect the time to use the step keys to be governed by the number of keystrokes which must be used to move the cursor to the target. Since the keys can only move the cursor vertically or horizontally, the number of keystrokes is $D_x/0.456 + D_y/0.246$, where D_x and D_y are the horizontal and vertical components of distance to the target; 0.456 cm is the size of a vertical step and 0.246

cm is the size of a horizontal step. Hence positioning time should be

$$T_{pos} = K_0 + C\,(D_x/0.456 + D_y/0.246). \qquad (6.6)$$

This equation with $K_0 = 1.20$ sec and $C = 0.052$ sec/keystroke has a standard error of 0.54 sec and explains 84% of the variance of the means.

Since the tapping rate is around 0.15 sec/keystroke, C is much too fast to be identified with the pressing of a key. It is also too fast to be identified with the 0.067 sec/keystroke automatic repetition mode. Plate 6.7 shows positioning time plotted against the predicted number of keystrokes. The long solid line is Equation 6.6 with the above parameters. The figure shows that positioning time is linear with the number of keystrokes until the predicted number of keystrokes becomes large (that is, the distance to the target is long). In these cases the user often has the opportunity to reduce positioning time by using the HOME key.

Fitting Equation 6.6 to the first part of the graph $(D_x/0.456 + D_y/0.246 < 40)$ gives

$$T_{pos} = 0.98 + 0.074\,(D_x/0.456 + D_y/0.246).$$

The equation, indicated as a short solid line on the figure, has a standard error of 0.18 sec and explains 95% of the variance in the means. The reasonable slope of 0.074 sec/keystroke shows that the 0.067 sec/keystroke automatic repetition feature was heavily used.

Text Keys

The text keys present the user on most trials with a choice of methods to reach the target. For example, he might press the PARAGRAPH key repeatedly until the cursor has moved to the paragraph containing the target paragraph. He could then press the LINE key repeatedly until it is on the target line, then use the WORD key to bring it over to the target. Or he might use the PARAGRAPH key to move to the paragraph after the target, then holding, the REVERSE key down, use the LINE key to back up to the line after the target line. And finally, using REVERSE and WORD, back up until he hits the target. In fact, there are 26 different methods for moving the cursor to the target, although only a subset will be possible in a given situation. The fastest method will depend on where the target is located relative to the starting position and the boundaries of surrounding lines and paragraphs.

A reasonable hypothesis would be that positioning time is proportional to the number of keystrokes and that for well practiced subjects the number of keystrokes will be the minimum necessary. To

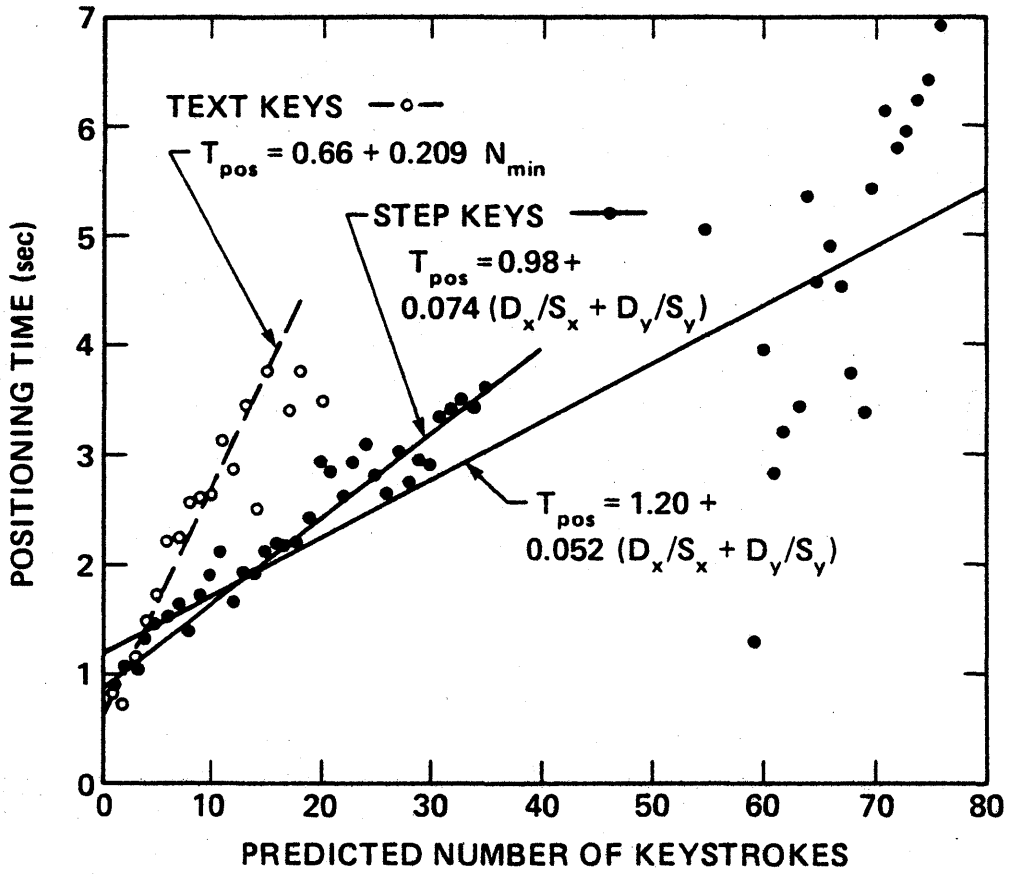


Plate 6.7. Positioning time for key devices as a function of predicted number of keystrokes.

test this hypothesis each trial was analyzed to determine the minimum number of keystrokes N_{min} necessary to hit the target. The average positioning time as a function of N_{min} is plotted as the open circles in Plate 6.7. A least squares fit gives

$$T_{pos} = 0.66 + 0.209\ N_{min}.$$

The standard error is 0.24 sec and the equation explains 89% of the variance of the means. The keystroke rate of 0.209 sec/keystroke is very reasonable, being approximately equal to the typing rate for random words (Devoe, 1967). Evidently, the automatic repetition mode was little used. Examination of some statistics on the minimum numbers of keystrokes for each trial shows there was little need for it. For one thing, an average of only six keystrokes was necessary for the text keys to locate a target word. Ten or fewer keystrokes were sufficient for over 90% of the targets. For another, these keystrokes were distributed across several keys, further limiting opportunities to use the repetition mode. The PARAGRAPH key was needed on 48% on the trials, the LINE key on 85%, the word key on 83%, and the REVERSE key on 81%.

Comparison of Devices

Table 6.3 summarizes the models, the standard errors of the fit, and the percentage of variance between the means explained by the model.

The match of the Fitts's Law slope to the roughly $K \simeq 0.1$ sec/bit constant observed in other hand movement and manual control studies means that positioning time is apparently limited by central information processing capacities of the eye-hand guidance system (cf. Welford, 1968; Glencross, 1977). Taking $K = 0.08$ sec/bit as the most likely minimum value for a similar movement task, and $K_0 = 1$ sec as a typical value observed in this experiment, it would seem unlikely that a continuous movement device could be developed whose positioning time is less than $1 + 0.08 \log_2(D/S + 0.5)$ sec (unless it can somehow reduce the information which must be centrally processed), although something might be done to reduce the value of K_0. If this is true, then an optimal device would be expected to be no more than about 5% faster than the mouse in the extreme case of 1 character targets 16 cm distant ($1 + 0.095 \log_2[(16/1) + 0.5] = 1.38$ sec vs. $1 + 0.08 \log_2[(16/1) + 0.5] = 1.32$ sec). Typical differences would be much less. By comparison in this same case, the joystick (in this experiment) is 83% slower than the optimal device, the text keys 107% slower, and the step keys 239% slower. Even if K_0 were zero, the mouse would still be only 23% slower than the minimum. While devices might be built which improve on the

TABLE 6.3

SUMMARY OF MODELS FOR POSITIONING TIME (T_{pos})

Device	Model (times in sec)	s_e	R^2
Mouse	$T_{pos} = 1.03 + 0.096 \log_2 (D/S + 0.5)$	0.07	0.83
Joystick	$T_{pos} = 0.99 + 0.220 \log_2 (D/S + 0.5)$[a]	0.13	0.89
	$T_{pos} = K_D + 0.1 \log_2 (D/S + 0.5)$[b]	0.07	—
Step Keys	$T_{pos} = 1.20 + 0.052 (D_x/S_x + D_y/S_y)$[c]	0.54	0.84
	$T_{pos} = 0.98 + 0.074 (D_x/S_x + D_y/S_y)$[d]	0.18	0.95
Text Keys	$T_{pos} = 0.66 + 0.209 N_{min}$	0.24	0.89

[a] Least squares fit to all data points.
[b] Fitting a separate line with slope .1 sec/bit for each distance.
[c] Least squares fit to all data points.
[d] Fit for number of keystrokes ($D_x/S_x + D_y/S_y$) < 40, where HOME key unlikely to be used.

mouse's homing time, error rate, or ability for fine movement, it is unlikely their positioning times will be significantly faster.

This maximum information processing capacity probably explains the lack of any significant difference in positioning time between the lightpen and the lightgun in Goodwin's experiment. Both are probably Fitts's Law devices, so both can be expected to have the same maximum 0.1 sec/bit rate as the mouse (if they are optimized with respect to control/display ratio and any other relevant variables).

In interpreting these results, highly favorable to the mouse, some qualifications are in order. Of the four devices, the mouse is clearly the most "compatible" for this task (cf. Poulton, 1974; Chapter 16), meaning less mental translation is needed to map intended motion of the cursor into motor movement of the hands than for the other devices. Thus it would be expected to be easier to use, put lower cognitive load on the user, and have lower error rates. There are, however, limits to its compatibility. Inexperienced users are often bewildered about what to do when they run the mouse into the side of the keyboard trying to move the cursor across the screen. They need to be told that their mice can simply be picked up and deposited at a more convenient place on the table without affecting the cursor. Even experienced users are surprised at the results when they hold their mice backwards or sideways.

The greatest difficulty with the mouse for text-editing occurs with small targets. Punctuation marks such as a period are considerably smaller than an average character. The error rate for the mouse, which was already up to 9% for one character targets, would be even higher for these sorts of targets.

6.4 SUMMARY AND CONCLUSION

Of the four devices tested the mouse is clearly the superior device for text selection on a CRT:

1. The positioning time of the mouse is significantly faster than that of the other devices. This is true overall and at every distance and size combination save for single character targets.

2. The error rate of the mouse is significantly lower than that of the other devices.

3. The rate of movement of the mouse is nearly maximal with respect to the information processing capabilities of the eye-hand guidance system.

As a group the continuous movement devices are superior in both speed and error-rate. For the continuous movement devices, positioning time is given by Fitts's Law. For the key devices it is proportional to the number of keystrokes.

7

Simulating User Performance on a Display Editor

This chapter carries the GOMS theory of performance in the manuscript editing task three steps farther. First, it extends the analysis to behavior with Editor Y, a CRT display-based editor. Second, it formalizes the model into a running computer simulation program. Third, it introduces stochastic decision making into the model in order to predict the distribution of decisions the user will make and the distributions of times it will take him to perform the modifications.

7.1 ANALYSIS OF THE TASK ENVIRONMENT

Within broad limits, the user will seek to behave in rational ways constrained by the structure of his environment. This is the Principle of Limited Rationality (Newell and Simon, 1972). To predict the behavior of a person doing a task, it suggests, one must first analyze the task environment to discover the environment's constraints. In the following analysis, these constraints are reflected: first, in the limited ways in which the different objects in the environment can interact; second, in the structure of the space of editing tasks considered. These two classes of environmental influences will now be discussed. A third classes of reflections of the task environment show up in the structure of the model of the user. They will be considered in a later section.

Transactions

A user of Editor Y sits before a CRT display with a keyboard for character input and a mouse for pointing. There are four main entities in this environment, (1) the marked up paper manuscript, (2) the user, (3) the computer editing system with its keyboard and mouse, and (4) the display. We can describe the environment in terms of a set of *transactions* between these entities (see Plate 7.1). The user consults the manuscript to find the next task; he seeks from it more information about the task under way. The user issues commands to the editor. The editor changes the display. The user consults the display for the location of a piece of text. Two of these entities, the user and the editor, are active, able to initiate transactions. Two entities, the manuscript and the display, are passive, speaking only when spoken to.

By listing the transactions available between entities in the environment, we can give a brief description of those entities. This description is not intended describe them completely, but only those aspects which affect the structure of the model.

User → Editor Transactions

(*I:*)
(*R:*)
(*D:*)
(*J:* (*is* #*Task*))
(*Key:* (*is* #*Text*))
(*Scroll:* (*is* #*Task*))
(*Select:*)

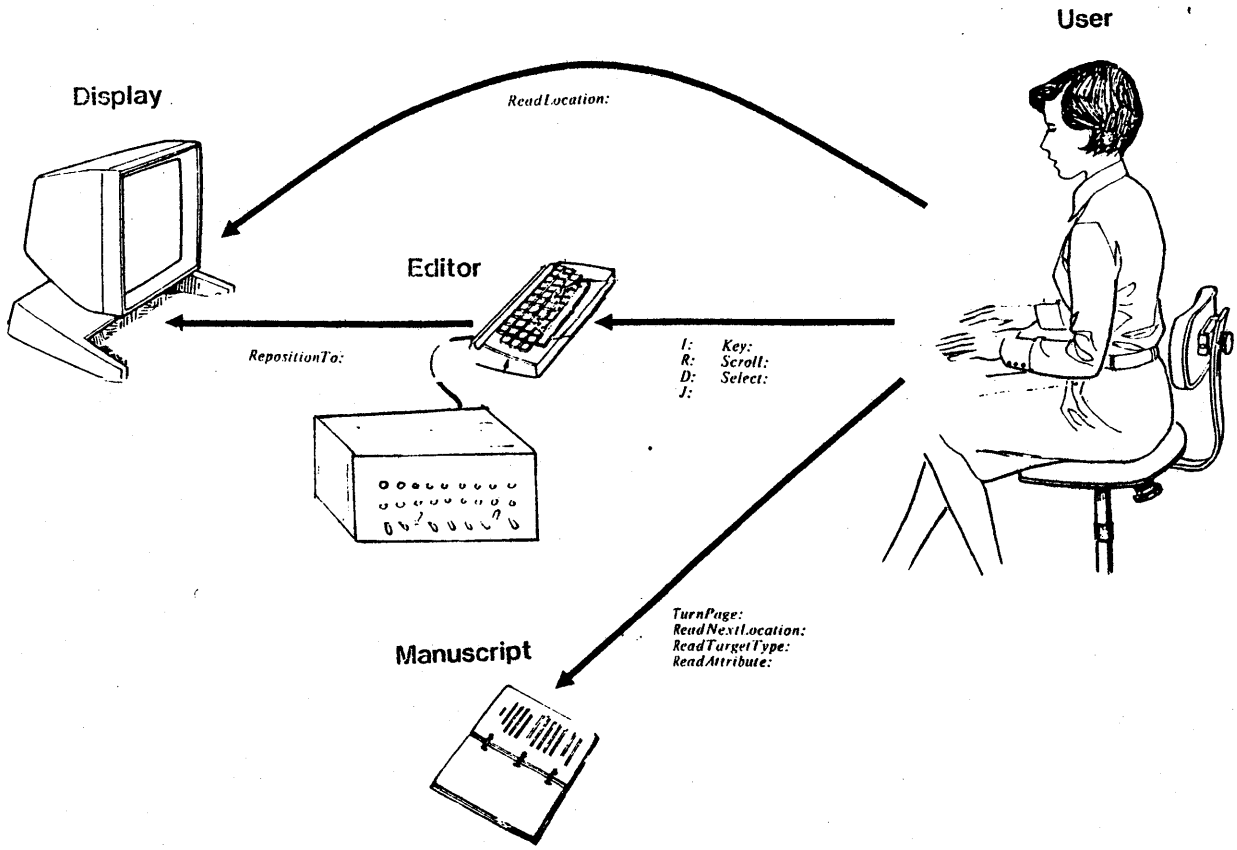


Plate 7.1. Transactions between entities in the task environment.

These transactions represent schematically the commands available on the editor. *I:* is the insertion command. It represents the initiation of the command (by typing the letter "I" in Editor Y) and the termination of the command (by typing < ESC > in Editor Y). The typing of text to be inserted is represented by the *Key:* transaction. *R:* is the replace command, *D:* the deletion command. *J:* the jump command repositions the display with the first line to match a typed-in string at the top. *Scroll:* repositions the display so that a location pointed to with the mouse is at the top or the bottom. *Select:* instructs the system to mark some piece of text for selection. Hitting one button on the mouse selects the character nearest the cursor, hitting another selects the nearest word. Segments of text are selected by pointing to the beginning, pressing a button, pointing to the end, pressing another button. The expression (*Scroll:* (*is #Task*)) means generically "Scroll down to some task".

Editor → Display Transactions

(*RepositionTo:* (*is #Task*))

Whenever scrolling or jumps occur the editor repositions the text on the display.

User → Display Transactions

(*ReadLocation:* (*is #Task*))
→ (*is #MainPart*) *or*
(*is #NearBottom*) *or*
(*is #Offscreen*)

This is how looking at the display screen to find the task is described. Knowing the task for which he is looking, the user looks at the face of the display. We describe this as sending a message *ReadLocation* to the display. The display makes a reply giving the location of the task. The exact location on the screen is of little use in predicting his performance, but what is important is whether the task is in the main part of the screen, the bottom part, or not on the screen at all. Notice that changes in the location of the task will be reflected in changes to the internal state of the display entity alone. The user may have an internal memory of where the task was that may or may not correspond with the display's state.

User → Manuscript Transactions

(*TurnPage:* (*is #Direction*))
→ *OK* or

NoMorePages

(*ReadNextLocation:* (*is #Task*))
→ (*is #Task*) or
NoMoreTasksThisPage
(*ReadTargetType:* (*is #Task*))
→ *InsertionPoint* or
(*is #Character*) or
(*is #Work*) or
(*is #Textseg*)
(*ReadAttribute:* (*is #Task*)
(*is #TaskAttribute*))
→ (*is #Attribute*)

The user can turn the pages of a manuscript forward or backward. If he tries to turn over the last page, he discovers immediately, of course, that there are no more pages. A user can look for the task on a page which follows a certain task, he can note what sort of target it is he must select, or he can read various attributes of a task, such as the new text that is to be inserted. In modeling, the manuscript, the user, the editor, and the display will each be treated as a separate process with all interaction occurring through only these transactions.

The Space of Tasks

Now we shall consider how to describe the space of editing tasks in the domain of the system. Consider a fragment of a typical marked up manuscript (Plate 7.2). For the task labeled A2, the instructions marked on the manuscript indicate the character "a" is to be inserted as a word between "involved" and "necessary." Symbolically we might describe this task as a tree

A2 = (*#Insertion*
(*Function:* *Insert*)
(*InsertionPoint:* *InsertionPoint2*)
(*NewText:* *Character1*)
(*RelTaskNo:* 2)
(*LineNo:* 12)
(*Previous:* *A1*)
(*Next:* *A3*)).

CHAPTER 7

Chapter I: INTRODUCTION

While the official, chartered purpose of this Subcommittee on Data Base Management systems is to investigate the potential for standardization in the area of data base management systems, a necessary first step of the work of the Subcommittee has been the development of a set of requirements for effective data base management systems. These requirements have emerged as the work of the Subcommittee manifested proceeded and have themselves in the form of a generalized model for the description of data base management systems. As no existing or proposed implementation of a data base management system completely satisfies these requirements nor comprises all of the concepts involved, necessary discussion of standards is an explanation of this model. The bulk of this Report provides such an explanation.

As a preliminary, it is ~~within reason to explain what events led to the preparation of this document.~~ Among the responsibilities of the Specifications Planning and Requirements Task Force of the Ad Hoc Marketing Committee for Computers and Information Processing is the generation of recommendations for action by the parent Task Force on appropriate areas for the initiation of specifications development efforts. For some time, starting in about 1969, the task force has been aware that data base management systems are becoming central elements of information processing systems, and that there is less than full agreement in the community on appropriate design. In addition to the existence of a number of implementations of such systems, a list that continues to grow, there are several documents generated out of the collective wisdom of some segment of the information processing community which are either proposals for specific systems (SMITH 1971) or more general statements of requirements (JAYME 1970), (HO 1971). As is well known, there is a debate in the community on whether existing and proposed implementations meet the indicated requirements, or whether the requirements as drawn are all really necessary ~~and entirely useful.~~ Further, there have been serious questions about the economics of systems meeting all the stated requirements.

Plate 7.2. Sample page of manuscript

where

Character1
= (#*Character*
(*TextType:* *Character*)
(*Boundary:* *Word*)
(*Length:* 1)).

The expression may be read "A2 is an instance of an insertion with a Function Insert and an InsertionPoint InsertionPoint2 and ...." We shall limit our consideration to five of the most common instruction types: (1) instructions to insert some new text, (2) instructions to delete some old text, (3) instructions to replace some old text by some new text, (4) instructions to move some text somewhere, and (5) instructions to transpose two adjacent pieces of text. The page in Plate 7.2 contains four of these types. These instructions, together with the definition of their subparts, define a space of editing task. Formally that definition is given in Table 7.1. The symbolic representation for the editing modifications in Plate 7.2 is given in Table 7.2.

7.2 MODEL OF THE USER

The model to be proposed is an extention of the model of Chapter 4. In order to extend the model to a display-oriented editor the behavior of four users was videotaped and analyzed. One of these was used for detailed study, the other three mainly for comparison.

Method

Four users were videotaped using Editor Y on two standard manuscript editing tasks. All used the system daily in their normal work. Two were programmers and two were non-programmers. One of the programmers and one of the non-programmers were fast typists (86 and 73 words per minute), the other two slow typists (39 and 36 words per minute). The protocol chosen for detailed analysis belonged to one of the fast typists.

The users were presented with two marked up manuscripts, each containing 33 tasks. There were five types of tasks: insertions, deletions, replacements, moves, and transpositions. Text arguments were approximately 1, 4, 16, 64, or 512 characters long and were varied in their boundaries, whether within a word, on a word boundary, within a line, across two lines, or across two pages.

TABLE 7.1

DESCRIPTION OF THE SPACE OF EDITING TASKS WITHIN CAPABILITY OF MODEL

```
#Task ()
    isOneOf     ((is    #Deletion)
                 (is    #Insertion)
                 (is    #Replacement)
                 (is    #Move)
                 (is    #Transposition)) ;

#BasicTask (TaskNo: RelTaskNo: LineNo: Function:)
    hasParts    ((TaskNo:       (is  #Atom))
                 (RelTaskNo:    (is  #Integer))
                 (LineNo:       (is  #Integer))
                 (Function:     (is  #EditFunction))) ;

#Deletion (Function: OldText:)
    isA         #BasicTask
    hasParts    ((Function:     Delete)
                 (OldText:      (is  #TextInMs))) ;

#Insertion (Function: InsertionPoint: NewText:)
    isA         #BasicTask
    hasParts    ((Function:     Insert)
                 (InsertionPoint:
                                (is  #PlaceInMs))
                 (NewText:      (is  #Text))) ;

#Replacement (Function: NewText: OldText:)
    isA         #BasicTask
    hasParts    ((Function:     Replace)
                 (NewText:      (is  #Text))
                 (OldText:      (is  #TextInMs))) ;

#Move (Function: OldText: InsertionPoint:)
    isA         #BasicTask
    hasParts    ((Function:     Move)
                 (OldText:      (is  #TextInMs))
                 (InsertionPoint:
                                (is  #PlaceInMs))) ;

#Transposition (Function: LeftText: RightText:)
    isA         #BasicTask
    hasParts    ((Function:     Transpose)
                 (LeftText:     (is  #TextInMs))
                 (RightText:    (is  #TextInMs))) ;

#TextInMs ()
    IsOneOf     ((is    #WordInMs)
                 (is    #CharacterInMs)
                 (is    #TextsegInMs)) ;

#WordInMs (Location:)
isA #Word
    hasParts    ((Location:     (is  #PlaceInMs))) ;

#CharacterInMs (Location:)
    isA         #Character
    hasParts    ((Location:     (is  #PlaceInMs))) ;

#TextsegInMs (Location:)
    isA         #Textseg
    hasParts    ((StartLoc:     (is  #PlaceInMs))
                 (EndLoc:       (is  #PlaceInMs))) ;

#Text ()
    isOneOf     ((is #Word)
                 (is #Character)
                 (is #Textseg)) ;

#Word (TextType: Boundary: Length:)
    isA         #Text
    hasParts    ((TextType:     Word)
                 (Boundary:     Word)
                 (Length:       (is  #Integer))) ;

#Character (TextType: Boundary: Length:)
    isA         #Text
    hasParts    ((TextType:     Character)
                 (Boundary:     (is  #CharacterBoundary))
                 (Length:       1)) ;

#Textseg (TextType: Boundary:)
    isA         #Text
    hasParts    ((TextType:     Textseg)
                 (Length:       (is  #Integer))
                 (Boundary:     (is  #TextsegBoundary))) ;

#CharacterBoundary ()
    isOneOf     (InWord  Word) ;

#TextsegBoundary ()
    isOneOf     (Line SplitLines SplitPages) ;

#Bounds (Start: End:)
    hasParts    ((Start:        (is  #PlaceInMs))
                 (End:          (is  #PlaceInMs))) ;
```

TABLE 7.2

SYMBOLIC REPRESENTATION OF MANUSCRIPT FRAGMENT IN FIGURE 2

```
Page1
  = (#Page
        (Pageno:        1)
        (Tasks:         (*list A1 A2 A3 A4 A5)
        (Next:          Page2)) ;

A1
  = (#Move
        (Function:      Move)
        (OldText:       TextsegInMs1)
        (InsertionPoint: InsertionPoint1)
        (RelTaskNo:     1)
        (LineNo:        8)
        (Next:          A2)) ;

TextsegInMs1
  = (#TextsegInMs
        (StartLoc:      PlaceInMs8a)
        (EndLoc:        PlaceInMs8b)
        (TextType:      Textseg)
        (Length:        4)
        (Boundary:      Line)) ;

A2
  = (#Insertion
        (Function:      Insert)
        (InsertionPoint: InsertionPoint2)
        (NewText:       Character1)
        (RelTaskNo:     2)
        (LineNo:        12)
        (Previous:      A1)
        (Next:          A3)) ;

Character1
  = (#Character
        (TextType:      Character)
        (Boundary:      Word)
        (Length:        1)) ;

A3
  = (#Insertion
        (Function:      Insert)
        (InsertionPoint: InsertionPoint3)
        (NewText:       Textseg1)
        (RelTaskNo:     A3)
        (LineNo:        12)
        (Previous:      A2)
        (Next:          A4)) ;

A4
  = (#Replacement
        (Function:      Replace)
        (NewText:       Textseg2)
        (OldText:       TextsegInMs2)
        (RelTaskNo:     4)
        (LineNo:        16)
        (Previous:      A3)
        (Next:          A5)) ;

Textseg2
  = (#Textseg
        (TextType:      Textseg)
        (Length:        79)
        (Boundary:      SplitLines)) ;

TextsegInMs2
  = (#TextsegInMs
        (StartLoc:      PlaceInMs9a)
        (EndLoc:        PlaceInMs9b)
        (TextType:      Textseg)
        (Length:        21)
        (Boundary:      SplitPages)) ;

A5
  = (#Deletion
        (Function:      Delete)
        (OldText:       TextsegInMs3)
        (RelTaskNo:     5)
        (LineNo:        33)
        (Previous:      A4)
        (Next:          NIL)) ;

TextsegInMs3
  = (#TextsegInMs
        (StartLoc:      PlaceInMs10a)
        (EndLoc:        PlaceInMs10b)
        (TextType:      Textseg)
        (Length:        71)
        (Boundary:      SplitLines)) ;

Textseg1
  = (#Textseg
        (TextType:      Textseg)
        (Length:        19)
        (Boundary:      Line));
```

Outline of the Model

According to the model of Chapter 4, a user may be represented as a 4-tuple **< G, O, M, S >**, where

G is a set of *goals* defining a state of affairs to be achieved,

O is a set of *operators* which specify the possible set of changes to the user's memory or to the task environment,

M is a set of *methods*, that is; sequences of goals and operators associated with the intended achievement of particular goals,

S is a set of *selection rules* for choosing among the multiple methods which may be applicable to a goal.

For the present model the elements of these sets are enumerated in Table 7.3. In that table and in what follows we shall follow the convention that items beginning with a * are pieces of information that must at some time be kept in the user's short term memory.

Goals. The goals of Table 7.3 form a subgoal hierarchy as pictured in Figure 7.3. The user is assumed to have an overall goal *EditMs* to edit the manuscript. Since the instructions written on the manuscript are broken down into individual tasks, he will also generate a set of subgoals *EditTask*. In order to do the editing required for one of these tasks, however, he must first find out what the task is by setting up the goal *GetTask*. The result of *GetTask* is one of the goals *Replace, Delete, Insert,* or *Move.* These goals are qualified by some reference to the new text to be inserted (**NewText*), the text to be deleted (**OldTextKey*), or the place where new text is to go (**InsertionPointKey*). The precise form in which the user has these pieces of knowledge represented in his head is not specified.

In order to select segments of text, the user employs *SelectTarget*, which uses the subgoals *PointToTarget* and *PointThere* to handle the details of pointing.

Operators. The user is assumed to have available to him a certain number of elementary behavioral acts. Two of these, *GetFromMs* and *GetFromDisp* have to do with getting information from the environment, the manuscript, and the display. Two of the operators describe the users actions with the mouse. *Point* is the act of causing the cursor on the display to move by means of moving the mouse until it is at a certain *ScreenPosition* described by *Key.* The argument **Bug?* is either "Bug"

TABLE 7.3

GOMS MODEL FOR EDITOR Y

Goals

(*EditMs*)
(*EditTask*)
(*GetTask*)
(*Insert *InsertionPointKey *NewText*)
(*Delete *OldTextKey*)
(*Replace *OldTextKey *NewText*)
(*Move *InsertionPointKey *OldTextKey*)
(*SelectTarget *PlaceInMs *WhichTarget *Target*)
(*PointToTarget *PlaceInMs *Target *Bug*)
(*PointThere *ScreenPosition *TargetType *Bug*)

Operators

(*GetFromMs DesiredInformation Attribute*)
(*GetFromDisp DesiredInformation Attribute *PlaceInMs*)
(*Scroll Place*)
(*Jump Place*)
(*Point ScreenPosition Key Bug?*)
(*I*)
(*D*)
(*R*)
(*Key NewText*)

Methods

OneAtATimeMethod
GDVMethod
ReadTaskInMsMethod
ICommandMethod
DCommandMethod
RCommandMethod
Delete-InsertMethod
ZeroInMethod
RoughPointMethod
CharPointMethod
WordPointMethod
TextsegPointMethod
InsertionPointMethod
PointWithoutScrollingMethod
ScrollAndPointMethod
JumpMethod

Selection Rules

RoughLocRule
TextsegRule
CharPointRule
WordPointRule
InsertionPointRule
Top2/3Rule
Bottom1/3Rule
OffScreenRule

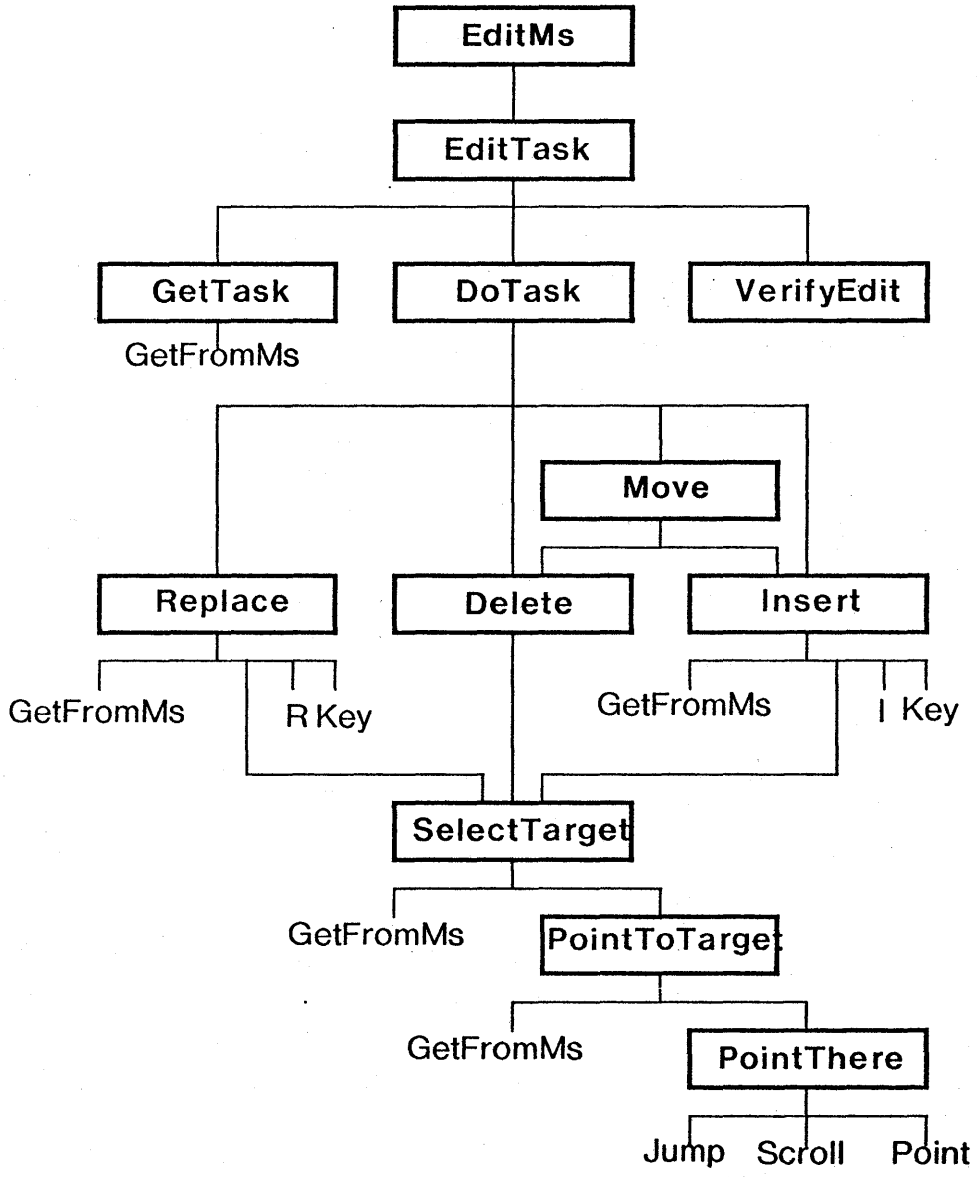


Figure 7.1. Hierarchy of goals for Table 7.1

meaning push a button on the mouse to signal the text under the cursor is selected or "Don't Bug." *Scroll* means to move the cursor to the edge of the screen and push a button causing all the text on the screen to move up or down. *Jump, I, D,* and *R* mean to issue those commands to the editor. *Key* means to type in a text argument. *VerifyEdit* is to check that the task was done correctly. *DoTask* is the operator which sets up as a goal the instruction obtained by *GetFromMs.*

Methods. An important part of the model is the means-ends analysis which sets out methods for the accomplishment of the various goals. Some of these methods will be presented here. Others will be deferred until later.

The method for accomplishing *EditMs* is simply to proceed one task at a time through the manuscript. This method, called the OneAtATimeMethod, can be written as follows:

(*OneAtATimeMethod*
    (until *****NoMorePages* **do** (*EditTask*))).

In terms of the simulation we can think of **NoMorePages* as a variable. If its value is T (if **NoMorePages* is true) the loop will exit. In terms of the user this device tests for the existence of some knowledge element (**NoMorePages TRUE*) in short term memory.

Another example of a method is that used to edit a single task (that is, to achieve the goal *EditTask*). The user first gets the task into memory, does it, then, about 40% of the time, verifies his modification to be sure it is correct. This method, the *GVDMethod,* can be written:

(*GDVMethod*
    (*GetTask*)
    (*DoTask*)
    (with-probability .4 **do** (*VerifyEdit*)))

The method for getting the task can be expressed directly in terms of the *GetFromMs* operator.

(*ReadInTaskMethod*
    (*GetFromMs* '**Task*))

The need to distinguish between a symbol itself and some other symbol associated with the first symbol unfortunately leads to the notational encumberance of prefixing single quotation mark to indicate the former. If we think of the *'d elements as representing the names of slots in short term memory, the above expression gives the operator *GetFromMs*

the name of the slot rather than the contents.

The method for achieving an Insert is slightly more complicated.

```
(ICommandMethod
    (if ~ *InsertionPointKey
        then (GetFromMs '*InsertionPointKey))
    (SelectTarget *PlaceInMs 'InsertionPoint:
                                '*InsertionPointKey)
    (I)
    (if ~ *NewText then (GetFromMs '*NewText))
    (if *NewText ~= 'Default then (Key *NewText)))
```

If the user does not know where to make the insertion he looks over to the manuscript to find out. Then, with the mouse, he selects that place and issues the insertion command to the editor. If he cannot remember the text to be inserted he consults the manuscript. Finally, except in the special "Default" case where the text to be inserted is the argument to a previous command (for example the delete command) the user types in the new text. The methods for the other commands are similar.

Selection rules. When more than a single method is available, the model uses "SelectionRules" to choose among them. A simple example is the goal *PointToTarget.* In Editor Y, there are at least three major alternative methods for this goal: (1) to select a character, the user moves the mouse and presses the first button on it; (2) to select a word, he moves the mouse and pushes another button; (3) to select a text segment, he moves the mouse to point to the beginning of a the text segment, pushes a button, moves the mouse to point to the end of the segment, pushes a button. The corresponding selection rule set may be written:

```
(CharPointRule
    (if (is #Character *Target)
        then (CHOOSE CharPointMethod)))

(WordPointRule
    (if (is #Word *Target)
        then (CHOOSE WordPointMethod)))

(TextsegRule
    (if (is #Textseg *Target)
        then (CHOOSE TextsegPointMethod)))
```

Detailed Studies

While it was possible, above, to spell out the general outlines of a model, the settling of several issues depends on more detailed analysis.

Reasons for and frequency of GetFromMs. It is easily observed that the user often consults the manuscript several times during the course of a task. Under what conditions and how frequently will he consult the manuscript? And in particular, how can a method be written to describe the variable number of consultations the user requires in pointing to a target?

User S1's performance on the sixteen insertion tasks in the two manuscripts was examined with respect to the use of the *GetFromMs* operator. There were 40 instances of such an operator. Three different pieces of information the user sought to pick up from the manuscript could be identified: the location of the task, the operation to be performed, and the new text to be inserted. Of course, from a single look at the manuscript the user often picked up more than a single piece of information. Table 7.4 shows the distribution of reasons inferred for looking at the manuscript arranged by the number of characters in the new string. Each line in the table is a separate task. Each entry is the number of times the user looked at the manuscript for that reason in that task.

For example, on task A1, the user consulted the manuscript once at the beginning of the task. Since she proceeded to point at the target and then insert the new text without further consultations, she must have obtained all of these on the first look. It is therefore inferred that the location of the task, the operation to be performed, and the text to be inserted were all absorbed in that single consultation.

But on task A18, she looked once at the beginning of the task, then twice more at the manuscript before finally pointing to the target. Then she looked again before beginning to type in the new text. While typing she glanced back at the keyboard twice. From the first look she probably learned the approximate location of the task and the operation to be performed. On the second look she got another rough location of the target insertion point. On the third she learned the exact target position and on the fourth glance got the beginning of the text to be inserted. At this point she proceeded to type while watching the manuscript with small glances back to the display or keyboard to check for suspected errors or locate different keys (see Long, 1977). These glances are tallied in the "Type-watching" column. Since this sort of *GetFromMs* overlays the *Key* operations it will be hereafter ignored.

TABLE 7.4

REASON AND FREQUENCY FOR LOOKING AT MANUSCRIPT

Task	N chars.	Reason for Looking at Manuscript			Total	Type-watching
		Next Task (*GNT*)	Target Location (*GL*)	New Task (*GN*)		
A1		1	0	0	1	
A21		1	0	1	2	
B8	1	1	3	1	5	
B26		1	0	0	1	
A6		1	1	0	2	
A32		1	2	0	3	
B2	4.7	1	0	0	1	
B23		1	0	0	1	
A3		1	0	1	2	
A14		1	1	1	3	1
B1	18.2	1	1	1	3	
B16		1	1	2	4	1
A18		1	2	1	4	2
B6	75	1	1	1	3	2
A30		1	1	1	3	4
B10	522	1	0	0	1	7

The procedure for locating a target (*GL* column) is especially interesting. It typically goes as follows. First the user extracts a few words from the manuscript to use as a key. Either the words may be the exact target or some other words or characters she thinks may be heuristically useful. In either case she uses the mouse to point to the key. If the key is only a rough approximation to the target she does not bug the target, but looks over to the manuscript and repeats the procedure. Otherwise she bugs the target and moves on to the next step

TABLE 7.5

COMPARISON OF NUMBER OF *GL* OPERATIONS IN SEQUENCE WITH POISSON DISTRIBUTION

Number of *GL* Operators in Sequence	Frequency	
	Observed	Predicted $16\ (0.81^N e^{-0.81}/N!)$
0	7	7.1
1	6	5.8
2	2	2.3
3	1	0.6
4	0	0.1

of the task. This method of locating the target is here called the *ZeroInMethod* and may be described,

```
(ZeroInMethod
    (while (is #RoughKey *Target)
        do (PointToTarget *PlaceInMs *Target 'Don'tBug)
           (GetFromMs '*Target *WhichTarget)
        finally (PointToTarget *PlaceInMs *Target 'Bug))).
```

**Target* is the identifying key extracted from the manuscript by the user. **PlaceInMs* represents her memory for which task she is doing. **WhichTarget* identifies which of several possible targets she is considering (for example a move task has an InsertionPoint and an OldText).

Although it has not been possible to decide for any task how many times (*GetFromMs '*Target*) will be invoked in succession (and in an engineering analysis, such a prediction would usually need to be done in the absence of a particular manuscript) the numbers in Table 7.4 are well approximated by Poisson distribution of mean 0.81 (see Table 7.5). Hence the operator (*GetFromMs '*Target*) should be constructed so as to pick up rough location keys in such proportion that the number of iterations will be Poisson distributed.

TABLE 7.6

FREQUENCY WITH WHICH SUBJECT 1 EMPLOYED ALTERNATIVE METHODS FOR GOAL *POINT THERE* AS A FUNCTION OF DISTANCE OF TARGET FROM TOP OF SCREEN

Lines from Top of Screen		PointWithoutScrolling Method	ScrollAndPoint Method	Jump Method
On Screen	1-4	6		
	5-8	9		
	9-12	3		
	13-16	2	1	
	17-20	1	2	
	21-24		2	1
Off Screen	25-28		1	1
	29-32			
	33-36			1
	37-40			3

Scrolling. On a given task, the user might not reposition the screen at all. Or, he might "scroll" with the mouse to reposition the text up or down by a few lines on the screen (by moving the mouse to the left side of the screen and pushing a button the display can be made to jump a certain number of lines up or down). Or he might "jump" to a new location (by typing the letter J followed by some string of characters the text can be repositioned so that the first following instance of these characters will be at the top of the display). How can a set of selection rules be written which will predict the user's choice?

On each task the user has a choice of methods for adjusting the portion of the text file being displayed. In order to examine the way in which the user does scrolling, S1's performance on all of the tasks in the first manuscript were examined. For each task the number of lines from the top of the screen to the target were counted, and whether the user caused the text to be repositioned in the screen and how was recorded.

Table 7.6 shows the number of times the user adopted each of these methods as a function of the distance of the target from the top of the screen. The results for this subject may be simply expressed, if the target is in the top two-thirds of the screen, she does not reposition the screen; if the target is in the bottom third of the screen, she scrolls; if the target

is off the bottom of the screen, she uses the jump command:

```
(Top2/3Rule
     (if (is #MainPart *ScreenPosition)
         then (CHOOSE PointWithoutScrollingMethod)))

(Bottom1/3Rule
     (if (is #NearBottom *ScreenPosition)
         then (CHOOSE ScrollAndPointMethod)))

(OffScreenRule
     (if (is #OffScreen *ScreenPosition)
         then (CHOOSE JumpMethod)))
```

The MainPart of the screen is from 1 to 19 lines, the NearBottom part from 20 to 24 lines. All others are OffScreen.

This set of selection rules has the advantage that it makes clear the mechanism whereby the user makes his choice. It has the disadvantage that it demands knowing the state of the screen at some arbitrary point in the editing process. It would therefore be useful if there were a reasonably accurate set of selection rules which did not demand such a detailed knowledge of the situation.

The distance between two targets of the manuscript is easily determined by inspection of the manuscript alone. Table 7.7 shows the selection in methods as a function *Dy*, the number of lines from the last target. The selection rules are not as precise and make more errors, but have the considerable advantage that they can be determined without consideration of the details of how the screen will be updated. A set of selection rules based on *Dy* is given below.

```
(LittleDyRule
     (if Dy ≤ 16
         then (CHOOSE PointWithoutScrollingMethod)))

(MediumDyRule
     (if 16 > Dy ≤ 25
         then (CHOOSE ScrollAndPointMethod)))

(BigDyRule
     (if Dy > 25
         then (CHOOSE JumpMethod)))
```

TABLE 7.7

Frequency with which Users Employed Alternative Methods for Goal *PointThere* as a Function of Distance Between Tasks

Distance from	S1			S2		S3		S4	
Previous Task	PWSM[a]	SAPM[b]	JM[c]	PWSM	SAPM	PWSM	SAPM	PWSM	SAPM
0	12			11	2	13		12	1
1	16			15	1	15	1	15	1
4	11	2		10	3	8	5	7	6
16	3	6	7	4	11	2	13	2	13
32		7	7	1	6		7	3	4

[a] PointWithoutScrollingMethod
[b] ScrollAndPointMethod
[c] JumpMethod

How well do these two rules predict the user's method selection? Table 7.8 gives a number of statistics calculated for each rule. The hit rate is the percentage of time the rule was correct. The χ^2 statistic can be used to test the likelihood that these results could arise from chance. A normalized statistic which indexes the goodness-of-fit is the "correlation of attributes" $r = (\chi^2/N(k - 1))^{1/2}$. While the rule set based on the position of the target on the screen is better, the other rule set based on the distance between targets on the manuscript also makes a good showing.

This result encourages us to use the simpler distance between targets measure to examine the behavior of other users to see how stable these methods are across users. Table 7.7 shows the frequency with which the three other users used the differerent pointing methods. The main difference between these users and the first user is that they do not have the *JumpMethod* in their repertoires. All three switch from no movement to the *ScrollMethod* as the distance increases. The cross-over point varies from a distance of 4 lines between targets up to 11 lines. It would thus appear that the selection rule set

(*LittleDy2Rule*
  (**if** *Dy* < 8
    then (*CHOOSE PointWithoutScrollingMethod*)))

(*BigDy2Rule*
  (**if** *Dy* > 8
    then (*CHOOSE ScrollAndPointMethod*)))

is a reasonable rule set for those users who do not use the Jump method.

If the scrolling is the only means employed to move the text on the display, then almost independent of the distribution of tasks on the manuscript, the amount of scrolling will be determined by the length of the manuscript. From examination of the data from S2, S3, and S4, there are approximately 16 lines/scroll, hence the number of scrolls a user who does not use the *Jump* or *Find* command can be expected to do is given by

Total scrolls = 0.07 × *Total lines in manuscript.*

Table 7.9 shows the number of lines per scroll computed for individual users. Reasonable scrolling behavior may be approximated by having the model scroll 16 lines at a time.

TABLE 7.8

Goodness-of-Fit for Alternative Selection Rule Sets for Goal *PointThere*

Rule Set	User	Manuscript 1			Manuscript 2		
		Hit Rate	χ^2	r	Hit Rate	χ^2	r
Top1/3Rule Bottom1/3Rule OffScreenRule	S1	0.85	35.90[a]	0.74[b]			
BigDyRule LittleDyRule	S1	0.85	30.97	0.68	0.75	10.29	0.40
BigDy2Rule	S2	0.94	24.04	0.87	0.85	14.73	0.67
LittleDy2Rule	S3	0.94	24.92	0.87	0.85	14.73	0.67
	S4	0.88	17.84	0.74	0.91	21.22	0.80

[a] All χ^2 in table significantly different from 0, $p < 0.01$.
[b] All r in table significantly different from 0, $p < 0.05$.

TABLE 7.9

AVERAGE NUMBER OF LINES PER SCROLL FOR DIFFERENT USERS

User	Number of Lines per Scroll		
	Non-Error Tasks	Error Tasks	All Tasks
S1	16.3	15.5	17.5
S2	16.3	21.5	17.7
S3	12.7	7.9	11.6
S4	20.2	11.6	16.1
Average	16.4	14.1	15.7

Statement of the Model

Now the full model can be stated. The GOMS elements are given in Table 7.10. These are stated in a specialized notation which collects together methods, and selection rules for a goal. The concepts of the model and the full description of the two manuscripts is given in the Appendix.

A trace of the model for task A2 is given in Table 7.11. The elements preceeded by a ▸ symbol are operators. Those preceded by □ are transactions. Table 7.11 is only one of the possible sequences the model predicts for this task. Adopting the abbreviations

GNT = (*GetFromMs *Task*)
GL = (*GetFromMs *Target ...*)
GN = (*GetFromMs *NewText ...*)
PR = (*Point ... Don'tBug*)
PB = (*Point ... Bug*)

and writing down only the sequence of operator firings predicted by the model give the more compact version

GNT PR GL PB I 16 *VE.*

TABLE 7.10

METHODS AND SELECTION RULES FOR EDITOR Y

```
(GOAL EditMs ()
    (METHODS
        (OneAtATimeMethod
            (until *EndOfJob do (EditUt)))))
(GOAL EditTask ()
    (METHODS
        (GDVMethod (GetTask)
            (DoTask *Task)
            (with-probability .4 do (VerifyEdit)))))
(GOAL GetTask ()
    (METHODS
        (ReadTaskInMsMethod
            (GetFromMs '*Task))))
(GOAL Insert (*InsertionPointKey *NewText)
    (METHODS
        (ICommandMethod
            (if ~ *InsertionPointKey then (GetFromMs '*InsertionPointKey))
            (SelectTarget *PlaceInMs 'InsertionPoint: *InsertionPointKey)
            (I)
            (if ~ *NewText then (GetFromMs '*NewText))
            (if *NewText ~= 'Default then (Key *NewText)))))
(GOAL Delete (*OldTextKey)
    (METHODS
        (DCommandMethod
            (if ~ *OldTextKey then (GetFromMs *OldTextKey))
            (SelectTarget *PlaceInMs 'OldText: *OldTextKey)
            (D))))
(GOAL Replace (*OldTextKey *NewText)
    (METHODS
        (RCommandMethod
            (if ~ *OldTextKey then (GetFromMs '*OldTextKey))
            (SelectTarget *PlaceInMs 'OldText: *OldTextKey)
            (R)
            (if ~ *NewText then (GetFromMs '*NewText))
            (if *NewText ~= 'Default then (Key *NewText)))))
(GOAL Move (*InsertionPointKey *OldTextKey)
    (METHODS
        (Delete-InsertMethod
            (Delete *OldTextKey)
            (Insert *InsertionPointKey 'Default))))
(GOAL SelectTarget (*PlaceInMs *WhichTarget *Target)
    (METHODS
        (ZeroInMethod
            (while    (is #RoughKey *Target)
              do      (PointToTarget *PlaceInMs *Target 'Don'tBug)
                      (GetFromMs '*Target *WhichTarget)
              finally (PointToTarget *PlaceInMs *Target 'Bug)))))
```

TABLE 7.10
(continued)

```
(GOAL PointToTarget (*PlaceInMs *Target *Bug)
    (SELECTION-RULES
        (RoughLocRule
            (if (is #RoughKey *Target) then (CHOOSE RoughPointMethod)))
        (TextsegRule
            (if (is #Textseg *Target) then (CHOOSE TextsegPointMethod)))
        (CharPointRule
            (if (is #Character *Target) then (CHOOSE CharPointMethod)))
        (WordPointRule
            (if (is #Word *Target) then (CHOOSE WordPointMethod)))
        (InsertionPointRule
            (if (is #PlaceInMs *Target) then (CHOOSE InsertionPointMethod))))
    (METHODS
        (RoughPointMethod
            (GetFromDisp '*ScreenPosition *Target *PlaceInMs)
            (PointThere *ScreenPosition 'Word *Bug))
        (CharPointMethod
            (GetFromDisp '*ScreenPosition (the Location: of *Target) *PlaceInMs)
            (PointThere *ScreenPosition 'Character *Bug))
        (WordPointMethod
            (GetFromDisp '*ScreenPosition (the Location: of *Target) *PlaceInMs)
            (PointThere *ScreenPosition 'Word *Bug))
        (TextsegPointMethod
            (GetFromDisp '*ScreenPosition (the StartLoc: of Target) *PlaceInMs)
            (PointThere *ScreenPosition 'Character *Bug)
            (GetFromDisp '*ScreenPosition 'Character *Bug)
            (PointThere *ScreenPosition 'Character *Bug))
        (InsertionPointMethod
            (GetFromDisp '*ScreenPosition *Target *PlaceInMs)
            (PointThere *ScreenPosition 'Character *Bug))))
(GOAL PointThere (*ScreenPosition *TargetType *Bug)
    (SELECTION-RULES
        (Top2/3Rule
            (if (is #MainPart *ScreenPosition)
                then (CHOOSE PointWithoutScrollingMethod)))
        (Bottom1/3Rule
            (if (is #NearBottom *ScreenPosition)
                then (CHOOSE ScrollAndPointMethod)))
        (OffScreenRule
            (if (is #OffScreen *ScreenPosition)
                then (CHOOSE JumpMethod))))
    (METHODS
        (PointWithoutScrollingMethod
            (Point *ScreenPosition *TargetType *Bug))
        (ScrollAndPointMethod
            (Scroll *PlaceInMs)
            (Point *ScreenPosition *TargetType *Bug))
        (JumpMethod (Jump *PlaceInMs)
            (Point *ScreenPosition *TargetType *Bug))))
```

TABLE 7.11

TRACE OF SIMULATION MODEL FOR SEQUENCE 8 OF TASK A2

```
. (EditTask)
.   The only method is GDVMethod
.   Use GDVMethod
.   . (GetTask)
.   .   The only method is ReadTaskInMsMethod
.   .   Use ReadTaskInMsMethod
.   .   ▶ (GetFromMs *Task NIL)
                          □ (Manuscript1 (ReadNextLocation: A1)) → A2
                          □ (Manuscript1 (ReadAttribute: A2 Function:)) → Insert
                          □ (Manuscript1 (ReadAttribute: A2 NewText:)) → Character1
.   . (Insert (is #RoughKey) Character1)
.   .   The only method is ICommandMethod
.   .   Use ICommandMethod
.   .   . (SelectTarget A2 InsertionPoint: (is #RoughKey))
.   .   .   The only method is ZeroInMethod
.   .   .   Use ZeroInMethod
.   .   .   . (PointToTarget A2 (is #RoughKey) Don'tBug)
.   .   .   .   RoughLocRule recommends RoughPointMethod
.   .   .   .   Use RoughPointMethod
.   .   .   .   ▶ (GetFromDisp *ScreenPosition (is #RoughKey) A2)
                          □ (Display (ReadLocation: A2)) → (is #MainPart)
.   .   .   . (PointThere (is #MainPart) Word Don'tBug)
.   .   .   .   Top2/3Rule recommends PointWithoutScrollingMethod
.   .   .   .   Use PointWithoutScrollingMethod
.   .   .   .   ▶ (Point (is #MainPart) Word Don'tBug)
.   .   .   ▶ (GetFromMs *Target InsertionPoint:)
                          □ (Manuscript1 (ReadAttribute: A2 InsertionPoint:))
                              → InsertionPoint2
.   .   .   . (PointToTarget A2 InsertionPoint2 Bug)
.   .   .   .   InsertionPointRule recommends InsertionPointMethod
.   .   .   .   Use InsertionPointMethod
.   .   .   .   ▶ (GetFromDisp *ScreenPosition InsertionPoint2 A2)
                          □ (Display (ReadLocation: A2)) → (is #MainPart)
.   .   .   .   . (PointThere (is #MainPart) Character Bug)
.   .   .   .   .   Top2/3Rule recommends PointWithoutScrollingMethod
.   .   .   .   .   Use PointWithoutScrollingMethod
.   .   .   .   .   ▶ (Point (is #MainPart) Character Bug)
                          □ (User1 SelectionMade:) → NIL
                          □ (EditorY Bug:) → Ready
.   .   ▶ (I)
                          □ (EditorY (I:)) → Ready
.   .   ▶ (Key Character1)
                          □ (EditorY (Key: NewText)) → NewText
.   ▶ (VerifyEdit)
```

By running the simulation model several times the model can be used to make Monte Carlo predictions (Table 7.12) of:

1. the set of possible operator sequences the user will employ to do an editing task;

2. the relative frequency with which the different operator sequences will be employed;

3. the distribution of time for each of these sequences; (The table gives the means, standard deviation, 5th and 95th percentile limits.)

4. and finally, the distribution of times for the whole task.

By combining the predictions for each task, the model can be used to make predictions for an entire manuscript.

Estimation of Parameters

In order to make time predictions with the model it is necessary to make numerical estimates of several of its parameters. Because a model such as this one is to be used to predict user behavior in advance, and since there is no established methodology for optimizing a simulation model with such a large set of parameters, we do not seek that set of parameters which will optimize the fit of the model to the data. Rather, we attempt to make reasonable estimates of the parameters in advance, then test the predictions of the model against experimental evidence to see how well its predictions fared.

The estimates for the parameters are summarized in Table 7.13. The times for operators *GetFromMs* and *TurnPage* is taken from Chapter 4. The time to *Point* with the mouse is taken from Chapter 6. Since the data in that study were pointing times in isolation, the measurements were confirmed by comparing with measurements of S1 in the present editing task. The time for *Bug*ing with the mouse is set at one reaction time, 0.3 sec. Each of the commands *I*, *R*, and *D* is assumed to take about the same time for command invocation, plus additional time for typing in arguments. The command invocation is estimated to be like doing two *SPECIFY-CMD* operations in POET, since there is the command name to be typed and an < ESC > character at the end. The time for *VerifyEdit* is based on the time previously measured for Editor Y (based on 12 measurements). Keystroke time is based on an average

TABLE 7.12

PREDICTED TASK SEQUENCES AND SIMULATION STATISTICS FOR TASK A2
(TIMES IN SEC)

Seq No	Sequence	Freq	Mean	SD	5th-%tile	95th-%tile
1	*GNT PB I GN* 1	17	8.1	1.9	5.5	12.8
2	*GNT PB I* 1	15	5.2	1.5	3.1	7.4
3	*GNT PR GL PB I GN* 1	11	11.5	2.5	8.2	17.4
4	*GNT PB I* 1 *V*	10	7.0	2.4	4.5	12.1
5	*GNT PR GL PB I* 1	9	8.1	1.4	6.0	9.8
6	*GNT PR GL PR GL PB I GN* 1	8	15.6	2.7	11.7	20.1
7	*GNT PB I GN* 1 *V*	7	7.7	.9	7.0	9.6
8	*GNT PR GL PB I* 1 *V*	6	9.9	3.3	7.4	14.2
9	*GNT PR GL PR GL PB I* 1	5	12.0	3.3	9.2	17.1
10	*GNT PR GL PB I GN* 1 *V*	5	11.5	2.0	9.3	14.2
11	*GNT PR GL PR GL PR GL PB I* 1 *V*	2	17.2	1.8	15.9	18.5
12	*GNT PR GL PR GL PB I* 1 *V*	2	12.5	5.2	8.8	16.1
13	*GNT PR GL PR GL PR GL PB I* 1	2	17.6	.8	17.0	18.1
14	*GNT PR GL PR GL PB I GN* 1 *V*	1	13.1	-	13.1	13.1
	Overall	100	9.5	3.9	4.3	17.4

TABLE 7.13

PARAMETER ESTIMATES FOR MODEL

Parameter	Estimated Time (Sec) Mean	SD	Source
User Parameters			
GetFromMs	2.1	0.9	Card, Moran, and Newell (1976), Figure 5.2
TurnPage	2.1	1.4	Card, Moran, and Newell (1976), Figure 5.2
Scroll	2.6	1.4	Measurement of 10 instances
Point	1.7	1.3	Card, English, Burr (1977), Table 1
Bug	0.3	0.2	1 reaction time
I, D, R	0.8	0.6	Card, Moran, and Newell (1976), Figure 5.2 (two *SPECIFY.CMD's*)
VerifyEdit	1.1	1.0	Measurement of 12 instances
KeystrokeTime	0.127	0.064	Average of two typing tests, SD = 0.5 Mean
System Parameters			
I, D, R	1.1	0.4	Measured response time of 25 instances
J	1.0	1.0	Measured response time of 10 instances
Scroll	1.7	1.2	Measured response time of 10 instances

of two typing tests embedded in an editing exercize given to the user before the start of the experiment as a warmup. The standard deviation for the keystroke time is estimated by multiplying the mean time per keystroke by a typical coefficient of variation for typing of 0.5 (Kinkaid, 1975).

In order to estimate response time of the system, 25 *I:*, *D:*, or *R:* command invocations were measured. Since there was no obvious difference among the times taken by these commands, their measured times were pooled to give a common estimated time. Ten invocations of *J:* commands and ten of *Scrolls:* were also measured. These parameter estimates are necessarily sketchy, given the labor of obtaining them. But, in an exploratory model they suffice to give us a feeling for how well the model does in its prediction. Engineering use of such a model would most likely use equally rough parameter estimates.

7.3 PREDICTIONS OF MODEL

The model makes predictions for the sequence of operations to be used, the frequencies with which these sequences occur, the time for editing a manuscript, and the standard deviation of editing times. Comparison of these predictions with available data indicates that the model is about as good as running another subject.

Prediction of Operator Sequences

In order to predict the operator sequences for Manuscript1, 100 Monte Carlo runs were made for each task in the manuscript. A detailed analysis of the operators actually used by user S1 for the tasks in this manuscript was completed by hand from the videotapes. As illustrated in Table 7.12, the model predicts several different sequences which might be used by the user. Since on a given trial the user might have used any one of these sequences, the predicted sequence closest to the observed sequence was selected. The accuracy with which the model predicted the sequences was determined by first examining how well the closest sequence matched, and then checking whether these sequences occurred with the predicted frequency.

There is no accepted statistic with which to summarize goodness-of-match, so this was assessed in several ways. The simplest method is to just note how many sequences were matched exactly. For tasks done correctly by the user, the model produced an exact match half of the time. Tasks in which the user made an error were never matched exactly by the model.

Another method by which goodness-of-match can be assessed is to ignore the sequence properties and correlate the frequency with which operators are predicted to occur against the frequency with which they do occur. For each task the correlation between the sequence observed for S1 and the (best matching) predicted sequence was computed (see Table 7.14).

The model predicts well the frequencies of operators for tasks the user performed correctly. The average correlation is 0.94. All of the correlations for these tasks are significantly different from 0 at $p < 0.01$.

The model does not attempt special predictions for the case in which the user makes an error. It is therefore appropriate (and reassuring) that the average correlation falls to 0.61 and that only half are significantly different from 0 at the 0.01 level.

A third way by which goodness-of-match may be assessed is to count the number of insertions, deletions, and transpositions necessary to transform the predicted sequence into the observed sequence. This method makes use of the sequence properties of the prediction. The column labeled "Sequence" in Table 7.14 gives the nearest predicted sequence of operators for each task and the operations necessary to transform the predicted sequence into the closest observed one. For example, the closest simulation sequence predicted for the replacement task A31 was

GNT S PB PB R GN 19

But the observed sequence was

GNT J GL PB PB GN R 19

(The user used the *JumpMethod* instead of the *ScrollMethod* predicted and also got the new text from the manuscript before beginning the replace command, rather than after.) The differences between these two sequences are expressed

GNT (*S*)[*J GL*] *PB PB R*⇋*GN* 19.

The rounded brackets () mean "delete the enclosed elements," [] means "insert the enclosed elements"' and ⇋ means "transpose the adjacent elements."

The distance of one string from another can be indexed by computing

TABLE 7.14

GOODNESS-OF-FIT FOR PREDICTED SEQUENCES BEST MATCHING SEQUENCES OBSERVED FOR USER S1

Task	Sequence	Structure Comparisons r	β	Rank	Prob	Time(sec) Pred	Obs
Insertion							
A21	*GNT PR GT PB I* 1	1	0	5	0.47	9.6	5.9
A2	*GNT PR GT PB I* 1	1	0	5.5	0.40	9.6	13.7
A6	*TP GNT S PR GT PR GT PB I* 4 *V*	1	0	14	0.02	21.3	15.9
A32	*GNT PR GT PB I* 5 *V*	1	0	4	0.40	11.2	13.0
A3	*GNT PB I GN* 19 *V*	1	0	2.5	0.60	11.3	22.7
A14	*GNT PR GT PR GT PR GT PB I GN* 20 *V*	1	0	7	0.05	22.8	13.8
A18	*GNT (J) PR GT [S] PR GT PR GT PB I GN* 17 *V*	0.92	0.23	6	0.12	25.2	35.0
*A30	*TP GNT (PR)[S] GT [S] PB I GN* 510 *V [PB PB R* 1 *V S S]*	0.03	1.44	4	0.35	79.6	141.7
Replacement							
*A27	*GNT S [GT] PB (R) GN [I]* 1 *V*	0.54	0.86	9.5	0.22	13.3	12.6
*A15	*GNT S [GT] PB R* 1 *V [PB D V]*	0.75	1.00	6	0.35	11.2	17.1
A8	*GNT J [GT] PB PB R⇋GN* 4	0.91	0.43	1	1.00	12.7	20.1
A17	*GNT PB R* 4 *V*	1	0	5	0.34	7.3	17.6
A26	*GNT PB PB (R) GN [I]* 17	0.81	0.67	3	0.55	11.7	13.5
A31	*GNT (S)[J GT] PB PB R⇋GN* 19	0.72	0.71	1	1.00	16.2	26.3
A4	*GNT S (PK) GT PB [GT] PB R GN* 79 *V*	0.93	0.40	4	0.35	28.7	26.2
Deletion							
*A12	*GNT [J] PR GT (PR)[PB] GT PB D [PB GN* I] *V*	0.54	1.00	6	0.05	17.1	11.8
A23	*TP GNT PB D V*	1	0	3	0.53	8.9	13.1
A19	*TP GNT PR GT PR GT PB PB D V*	1	0	4	0.25	14.4	12.7
A24	*GNT PR GT PR GT PR GT PB [PB] D V*	0.97	0.20	8	0.02	18.2	21.5
A5	*GNT [S] PR GT PR GT PB PB D*	0.95	0.25	6.5	0.14	19.3	22.6
A28	*GNT PR GT PB PB D*	1	0	1	1.00	11.2	16.2
A29	*GNT (S PR)[GT J] GT (PR)[J] GT PR GT PB PB D V*	0.73	0.67	6.5	0.06	24.2	42.2
Transposition							
A20	*GNT J [GT] PR GT PB (PR GT) D (PB) I V*	0.92	0.50	2.5	0.66	19.6	17.4
*A7	*GNT PR GT PB D [D] (PB) I [I 3] V*	0.70	0.88	4	0.32	14.2	33.7
Move							
*A11	*GNT PR GT PB [PB] D PB I [I 1 V GT PB PB D] V*	0.89	1.25	3.5	0.56	14.2	19.6
*A1	*GNT PR GT PB PB [PB] D (PR GT) PB I V*	0.83	0.45	4	0.35	19.7	35.4
A16	*GNT PR GT PR GT PB [PB] D (PR GT PB) I V [PB D I]*	0.85	0.83	5	0.20	23.5	14.3
A10	*GNT PR GT PR GT PR GT PB PB D PB I V*	1	0	6	0.11	19.7	64.2

$$\beta = (I + D + S_{ID} + T)/S\ f_{i,pred}$$

where

I = Number of operators inserted
D = Number of operators deleted
S_{ID} = Number of non-contiguous sites where insertions or deletions occur.
T = Number of transpositions.

This index counts the number of changes per predicted operator firing necessary to transform the predicted sequence into the actual sequence. For correct tasks β has a fairly reasonable value of 0.23, about one change every four operators. For error tasks, however, $\beta = 0.98$ or one change for each operator.

There is no agreed threshold for these statistics that will certify the sequence predictions of the model. About the best that can be said is all the statistics seem to agree and indicate that the model makes good predictions for user S1's error-free performance. All agree and indicate the sequences predicted by the model are not descriptive of S1's performance for tasks on which she makes errors.

How well does the model predict the frequencies with which the alternative sequences are chosen? The column labelled "Prob" in Table 7.14 records the proportion of sequences occurring in the simulation with frequency less than or equal to the matched sequence. If we divide the tabulated value into quintiles, an equal number of matches should fall into each group. For the correct tasks, this is the case (Kolmogorov-Smirnov $D(21) = .22$, NS $p > 0.20$). For error tasks, there are more sequences in the low probability quintiles than would be expected by chance ($D(7) = 0.46$, $p < 0.05$). The model satisfactorally predicts sequence frequencies for S1's correct tasks; it does not predict sequence frequencies for her error tasks.

Prediction of Time Distributions

If the prediction of the sequences is acceptable, how well does the model fare in predicting the time necessary to perform the task? It should be remembered that these predictions are essentially zero parameter predictions. The first obvious comparison to make is between the observed times for S1 and the predicted times of the best match sequences of Table 7.14. This comparison is made in the column labeled

"BestMatch" of Table 7.15. As can be seen from the table, the predicted average time per task for correct tasks is about 43% too low. The root mean square of the error (RMSE) is about 36% of the average time/task. The correlation between predicted times and observed times is quite low ($r = 0.32$) for correct tasks, but very high ($r = 0.99$) for error tasks. The very high correlation for error tasks results partially because two tasks with a large number of characters to be typed are error tasks. Actually, in absolute terms, the error tasks are predicted less well than the correct tasks as can be seen from the larger RMSE for error tasks, some 48% of the mean predicted time.

The next comparison to make is between the times predicted by the model using the mean time for each operator. This comparison appears in the "Var Off". Surprisingly these predictions do just as well if not slightly better (in the sense that they have a higher correlation and lower RMSE) than the "best match" predictions.

Finally the predictions of the model using gamma-distributed random numbers with means and variances as given by the model parameters are compared with S1's observed performance in the column labeled "Var On." The main point of this model is to attempt a prediction of the distribution of task times. Since data was not available for S1 performing the same set of tasks several times, the model was used to predict the standard deviation of all the tasks taken together. Using the operator variances causes the model to increase its prediction of the standard deviation of the combinded set of tasks about 50%, to a value closer to but still only half of that observed for correct tasks.

One way of puting these predictions into perspective is to compare the time predictions with the predictions that could be made by simply timing how long it takes another user to do the same task. This comparison is made in the last three columns. Roughly, the model does about as well as experimentally running another user.

7.4 SUMMARY

The model is a reasonably good predictor of the sequence of operators for tasks done correctly by the user.

It exactly predicted the sequence half the time.

The predicted frequencies of operators correlated 0.94 with those observed.

The sequences occurred with the predicted frequencies.

TABLE 7.15

COMPARISON OF OBSERVED EDITING TIME BY SUBJECT S1 WITH PREDICTORS

	Observed	Model Prediction			Compared to Other Users		
	S_1	Best Match	Var Off	Var On	S_2	S_3	S_4
Correct Tasks (N=21)							
Mean time/task (sec)	22.6	15.8	14.9	14.6	19.0	13.7	55.5
SD time/task (sec)	13.8	6.1	4.6	6.6	17.3	18.4	62.5
RMSE (sec)		14.5	13.8	14.7	10.4	13.2	63.5
r		0.32	0.57	0.42	0.82	0.73	0.70
Error Tasks (N=7)							
Mean time/task (sec)	38.9	23.2	24.3	23.3	49.4	26.7	69.2
SD time/task (sec)	46.3	24.9	23.9	26.5	95.9	42.0	99.2
RMSE (sec)		26.2	25.7	26.3	48.8	15.0	58.6
r		0.98	0.99	0.98	0.98	0.98	0.98
All Tasks (N=28)							
Mean time/task (sec)	26.7	17.6	17.3	16.7	29.9	17.0	61.2
SD time/task (sec)	25.9	13.3	12.6	19.9	50.3	22.9	71.4
RMSE (sec)		18.2	17.5	18.3	26.4	13.7	62.3
r		0.86	0.92	0.89	0.94	0.94	0.81

On the average, it was necessary to make one adjustment to the sequence for each four operator firings.

It does not try to predict the sequence of operators which will ensue after an error and these error sequences are significantly different from the predicted sequences.

The mean time per task is predicted by the model to within about 50-60% of the observed values.

The observed standard deviation is about twice what the model predicted.

The root mean square of the error deviations between predicted an observed values is about 50-60% of mean time for a task. This compares to the roughly 30% obtained for models of the POET editor.

The simulation model predicts the times roughly as well as running a human subject.

These results are for essentially zero parameter predictions and with rather sketchy data. The fit of times could probably be improved somewhat by better quality data. For example the parameters based on the POET editor and other sorts of Editor Y tasks apparently underestimated the variance in this situation. On the other hand such simulations face limitations in their reliance on the summation rule for accumulating times, and other techniques for estimating times should be compared against these results to ensure there is sufficient gain from the simulations to repay their costs.

8

Conclusion

What can we predict about the behavior of the user in the manuscript editing task? What will be his sequence of actions? How long will it take him to perform them? What sorts of errors will he make? What do we now know about user performance in editing systems? Let us review the major findings of the thesis.

8.1 FINDINGS

The main findings of the thesis are classified according to topic in Table 8.1. The six studies described in the preceding chapters fall into a cyclical pattern of basic research and application.

[BASIC RESEARCH]

1. What performance differences are there between editors?

(Chapter 2: Benchmark study of editing systems)

Significant speed differences (up to about a factor of 3) exist between editors as a consequence of the number of keystrokes/task required and the use of a page display.

2A. When editors were compared by determining the times required by expert users to perform benchmark tasks: Letter Typing, Manuscript Editing, Table Typing, and Text Assembly, there was a greater difference among editors in the time to perform the Manuscript Editing benchmark than in the time to perform the other benchmarks.

2B. The RCG editing system (7 sec/modification) is faster than Editor Y (10 sec/mod), which is faster than TECO (11 sec/mod), which is faster than POET (17 sec/mod). All were much faster than using a typewriter.

2C. The ratio between the time required to edit the manuscript in Chapter 2 with the slowest and the fastest editor is 2.3. Making allowances for the likely existence of slightly better and slightly worse editors, it is probably correct to say the design of the editor can make a difference of a factor of 3 in time to edit a manuscript.

2D. There was a difference of about a factor of 1.2 between the best and the worst expert user within an editor.

2E. Teletype-like editors took twice as long as display-oriented editors (even though compared on CRT output devices of the same speed).

2F. On the benchmark task, errors increased the average time/modification by 9% (Range 0 to 24%).

2G. The speed of an editor is largely a function of the number of keystrokes required to do a task.

[APPLICATION]

II. Can a priori analysis yield significant predictions about editing performance?

(Chapter 3; Application: Predicting When an Editing System is Better Than a Typewriter)

It was possible on the basis of a simple a priori analysis, using data determined in Chapter 2, to predict the (surprising) outcome of a user experiment on text-editing.

3A. The time to type a text and the time to edit a text with WYLBUR are well fit with simple linear models.

3B. From these simple theories, the length of text at which it becomes faster to use an editor can be derived, as well as the density of modifications per line of text at which it becomes faster to retype the text instead of editing it.

3C. For a WYLBUR-like (POET-like) editor the modification crossover point is 0.7 modifications/line, hence if there is more than one modification every 1/0.7 = 1.4 lines, it is better to retype the text from scratch than to edit it.

3D. A user should be able to type about 16 new words in the time necessary to make a correction with WYLBUR, hence it is better to type at slow speed making few errors than to type at high speed with the intention of making corrections during a second pass.

3E. Most of the prediction error of the Constant Time per Modification model arose from errors in setting the parameters rather than from errors in the form of the model.

3F. Taylor approximations can be used to linearize the formulae for expected editing time in such a way that the sensitivity of predictions to error in input parameters can be determined and so that the blame for prediction errors can be localized.

[BASIC RESEARCH]

III. How is editing behavior organized? What effect does the grain of analysis have on the quality of the model?

(Chapter 4: The GOMS Model of Manuscript Editing)

Editing behavior is organized into cyclical operator sequences ("unit task cycles"). It can be described in terms of Goals, Operators, Methods, and Selection Rules. Refining the grain size of a model does not seem to affect much its predictive ability.

4A. User behavior in the manuscript editing task can be modeled in terms of a small set of Goals, Operators, Methods, and Selection Rules (GOMS). These elements can be used to produce models that predict the sequence of operators and the time to perform a modification.

4B. GOMS models can be written at several grain sizes: Constant time/modification models, task step models, argument-level models, and movement-level models.

4C. The GOMS model for POET did not improve in its ability to predict performance (of a single user) on a new task after the grain size was reduced below the 4 sec level (task step model).

4D. With two or three selection rules it is possible to predict method choices with 80-90% accuracy.

4E. The time for a given individual modification can be predicted by a GOMS model to within 20-35%.

4F. Errors (for one subject using POET) increased editing time by 25%.

4G. Individual differences were found among users in their preferences among methods of locating the target line.

[APPLICATION]

IV. How can rough performance measures be estimated for a system when it is in the conceptual design stage?

(Chapter 5; Application: Predicting User Performance at an Early Stage in System Design)

The unit task pattern discovered in Chapter 4 suggests a technique: enumerate the number of unit tasks and estimate the number of these required for a task.

5A. Since user performance on many editor-like systems is organized as a set of unit task cycles, the unit tasks of a system can be used at design time to predict the amount of time various user activities on the system can be expected to take.

5B. In order to carry through the above predictions, data on the relative number of various features per page were collected (Table 5.2).

5C. Using the number of unit tasks per task, the number of tasks per page, and the time per unit task it was possible to estimate the user time per page for different system configurations without knowing the details of the system.

5D. While unit tasks do interact with each other, attempting to capture this interaction by analysis at the next level of detail changed the results by only about 6%.

[BASIC RESEARCH]

V. Which is the best pointing device and why?

(Chapter 6; Evaluation of Mouse, Rate-controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT)

The mouse is fastest and most error free of the tested devices.

6A. The mouse (average positioning time 1.7 sec) is faster than the measured joystick (1.8 sec) is faster than the text keys (2.3 sec) is faster than the step keys (2.5 sec).

6B. The time to position a cursor with a key device (step keys or text keys) is proportional to the number of keystrokes necessary.

6C. The time to position a cursor with an analog pointing device (mouse or joystick) is proportional to $\log_2$ (*Distance/Target-size* + 0.5).

6D. The time to position a cursor with the mouse is

$$T_{pos} = 1 + 0.1 \log_2 (Distance/Target\text{-}size + 0.5) \text{ sec}$$

6E. The time to position the joystick measured was

$$T_{pos} = 1 + 0.2 \log_2 (Distance/Target\text{-}size + 0.5) \text{ sec.}$$

6F. The time to position the step keys is

$$T_{pos} = 1 + 0.074 (D_x/S_x + D_y/S_y),$$

where $(D_x/S_x + D_y/S_y)$ is the number of keystrokes to the target.

6G. The time to position the text keys is

$$T_{pos} = 0.7 + 0.21\ N_{min},$$

where N_{min} is the minimum number of keystrokes to the target.

6H. The speed with which pointing can be done with the mouse is close to the theoretical minimum.

[BASIC RESEARCH]

VI. How can the results be integrated into a simulation? How can stochastic uncertainty be added to the model of Chapter 4? Can the model be extended to a display editor?

(Chapter 7; Using GOMS to Simulate User Performance on a Display Editor)

A running computer simulation program was constructed incorporating many of the results of the thesis. This program, to date, is about as good at predicting editing performance as running another user.

7A. A GOMS model can be constructed and implemented down to the detail necessary for a running computer simulation for a display-based editor (Editor Y).

7B. The space of simple editing tasks is as given in Figure 6.3.

7C. The number of times a user looks at the manuscript before selecting a target with his pointing device was Poisson-distributed with mean 0.81 (based on one user in Editor Y).

7D. Users will scroll when the target is in the bottom third of the screen.

7E. Using two or three selection rules it was possible to predict the method for pointing at a target about 85% of the time.

7F. Users were quite similar in their choice of when to scroll. The main difference in screen movement methods found was whether or not the user ever used the jump command.

7G. Predictions of a GOMS simulation are about as good as predictions from examining the performance of another user in the same task.

7H. The simulator predicted sequences quite well (correlation between predicted and observed operator frequencies = 0.94; on the average, it was necessary to make one adjustment to the sequence for each four firings).

7I. The simulator predicted the mean time per task to within about 50-60% of observed values.

7J. The standard deviation of the time to do a task tasks was about twice what the model predicted.

In Table 8.1, the Basic Research-Application cycle is broken down still further. Some findings from the basic studies contribute measurements of constants such as performance time/operator. Other findings represent the results at theory building exercises. Finally, some findings have to do with attempts to verify theories and hypotheses. Table 8.1 shows that all are represented in the thesis.

The other side of the table lists three classes of applications: results can be used (1) to calculate predictions of behavior, (2) to make design decisions, and (3) to evaluate systems. The thesis has gone only so far as to make some sample applications of the results obtained.

8.2 ROUTINE COGNITIVE SKILL

Having laid out the results of our investigation into the manuscript editing task, it is time to ask how behavior in this task is related to, on the one hand, problem solving behavior and, on the other hand, skilled performance. Ultimately, the issue can be stated: how does the behavior of a user in the manuscript editing task arise from the nature of the user and his environment? The question is best discussed with respect to fundamental notions of man as symbol processing system.

The Internal Mechanisms of the User

Figure 8.1 depicts a model of the internal mechanisms of the user. The model comes from a wide range of basic experiments in psychology. Immediate sources for the figure are Newell and Simon (1972, Figure 4.1), Welford (1976, Figure 1.1), Sheridan Ferrell (1974,

TABLE 8.1

CLASSIFIED TABLE OF FINDINGS

TOPIC	BASIC RESEARCH			APPLICATION		
	Data Base	Theory	Verification	Calculation	Design	Evaluation
I. PERFORMANCE PARAMETERS						
Time	2A,2B,2C,2D 2E,2F.2G,4F	2F,2H,3A 3F,4A	3E,4E,7G -	3E,3D,3C,5C 3A	5A -	2A,2B -
Time Distribution	-	7J	7J	-	-	-
Errors	2G,4F	-	-	3D,3C	-	-
Operator Sequences	-	4A,7H	7H	-	-	-
Method Choices	-	4D,7E	4D,7E	-	-	-
II. EDITORS						
Comparisons	2B,2C,2E	-	-	-	-	2A,2B
WILBUR Editor	-	-	3A	-	-	-
POET Editor	2G,4H	4D	4C,4D,4E	3C,3D	-	2B
TECO Editor	2B	-	-	-	-	2B
Editor Y	2B,7C,7D -	7C,7E,7J 7A	7E,7G,7J -	5C -	- -	2B -
RCG Editor	2B	-	-	-	-	2B
Typewriter	2B	-	-	-	-	2B
III. POINTING DEVICES						
Mouse	5B	6C,6D,6I 6A	7E	-	-	-
Joystick	6A	6C,6E	-	-	-	-
Text Keys	6A	6B,6H	-	-	-	-
Step Keys	6A	6B,6G	-	-	-	-
IV. INDIVIDUAL DIFFERENCES	2A,7F,4H	-	-	-	-	-
V. MODELS						
Task Analysis	5B	4A,7B	7G,7A	-	5A	
Model Grain	-	4B	4C	-	-	-
Constant Time/UT Mods	-	2H,3F,4B	3E,3A,4E	5C,3B,3C	5A	-
UT Step-Level	-	4B	4E	5C	-	-
Average-Level	7C	4B	4F	-	-	-
Movement-Level	-	4B	4F	-	-	-
Keystroke	-	2F	2F	-	-	-

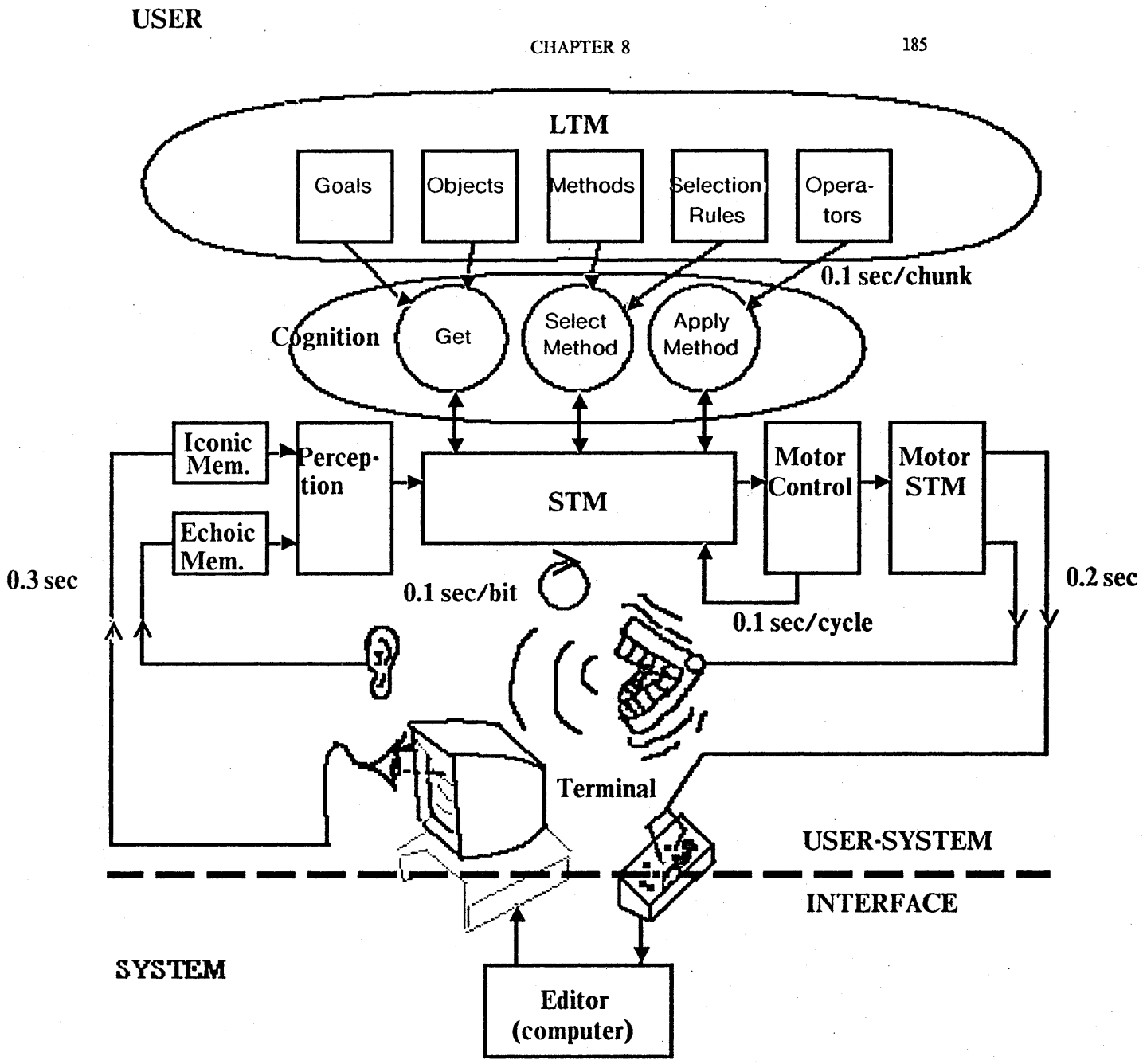


Figure 8.1. Internal mechanisms of the user

Figure 1.2), and results of the present thesis. The user has a single long term memory (LTM) of unlimited size which holds essentially permanent knowledge. There is also a short term memory (STM) of very limited size which holds the knowledge of the current task context; this represents the user's focus of attention. The user has organs of sensation with their associated buffer memories and some perceptual processing mechanism capable of rendering the sensations into symbols and depositing them in the short term memory. He also has motor organs and a motor control mechanism capable of executing limited motor programs.

The three major divisions of the central mechanisms deal respectively with perception, cognition, and the controlling of action (Welford, 1968; Welford 1976). Knowledge flows into the short term memory from both the external environment of the user and from his long term memory. Both of these flows are determined or moderated by the knowledge in the short term memory at each instant. Similarly, the user's initiation and control of external (motor) actions and of the addition of new knowledge to his long term memory is determined by the contents of working memory.

The perceptual component and the motor control component of the user are each capable of acting with a limited parallelism with the cognitive component. Thus a user may touch type, the perceptual mechanisms staying a word or two ahead of the motor control mechanism (Shaffer, 1976). Thus a child arranging a set of blocks by height may reach for the next block even as he is deciding where to put it (Young, 1976).

The cognitive component may be thought of as working with three logical processes: **GET**, **SELECT-METHOD**, and **EXECUTE**. These processes relate the goals, operators, methods, selection rules, and knowledge elements which describe the task, all in long term memory, to the instantaneous state of the task as represented in short term memory.

The Principle of Limited Rationality

The behavior of the user is the result of his attempt to achieve his goals by rationally acting in accord with the imperatives of the task environment (Simon, 1947; Simon, 1969; Newell and Simon, 1972). Thus, to change a word in the manuscript (goal) he employs the replace command (rational behavior), typing in the new word indicated on the manuscript (task environment). The behaviour of the user is also the result of limitations in his ability to behave rationally. To edit the manuscript (goal) he does not first read all of the indicated modifications at once then proceed to implement them (rational

behavior) because he cannot remember them (limitation).

Both the rationality of the user and the limits on that rationality are apparent in the model of the user in Figure 8.1. The rationality of the user is located in the analysis of the task environment that is embedded in the goals, operators, methods, and selection rules stored in long term memory. The limits of the user are in the form of processing or size limits on the parts of the figure. The principal ones may be expressed in very approximate numbers:

LTM size	∞	chunks[1]
STM size	7	chunks[2]
LTM -> STM transfer rate	0.1	sec/chunk?[3]
STM -> LTM transfer rate	5	sec/chunk[4]
Proprioceptive servo loop	0.1	sec/cycle[5]
Visual servo loop	0.5	sec/cycle[5]
Perception part	0.3	sec[5]
Motor part	0.2	sec[5]
Processing capacity	0.1	sec/bit[5]

Several of the results for this thesis can be derived directly from this simple view of the human information processing architecture:

(1) The user can do one main (cognitive) thing at a time—what the working memory determines at each instant—although he can overlap very limited amounts of perceptual and motor processing.

(2) His working memory is limited and overloading the working memory causes some immediate knowledge to be lost. Thus the user needs to consult the manuscript several times in the course of a modification.

(3) His knowledge in long term memory can only be retrieved if it can be accessed via the knowledge already in working memory. Thus, complete failures to recall relevant knowledge are possible, as are retrievals of inappropriate knowledge because the retrieval clues were inadequate.

(4) Adding knowledge to his long term memory takes much longer than retrieving knowledge. The user is forced to rely on his limited working memory to cope with rapidly changing task data.

(5) The information that the user encounters in the outside world must be encoded if it is to become usable and this can be done only in terms of the knowledge already available in the long term memory. Thus, the user can only learn new things in terms of knowledge he already has.

(6) The time/operator for a model at the individual user movement level (Model M0.5 in Chapter 4) can be expected to be on the order of 0.5 sec, the shortest time for a perceptual and motor cycle.

(7) The time to perform a keystroke is just the minimum time for a motor movement—around 0.2 sec.

(8) The movement times for the pointing devices cannot exceed about 0.2 sec/keystroke for the key devices or 0.1 sec/bit for the continuous movement devices.

Relation to Problem Solving

That the picture of the human processor is just basically the same picture that emerges from the study of problem solving can be seen by comparing Figure 8.1 to the description of a problem solver redrawn from Figure 4.1 of Newell and Simon (1972) in Figure 8.2. In both the human processor works by selecting and then applying methods from long term memory as controlled by the current contents of short term memory. Since the task in problem solving experiments is typically to solve a single problem rather than a series of tasks likein skilled performance, there is no **GET** in Figure 8.2. Also in Figure 8.1 the process **CHANGE REPRESENTATION** would be considered as just another method. The perceptual and motor control mechanisms are represented in Figure 8.1 in more detail, reflecting the important role these play in skilled performance.

The difference between manuscript editing and problem solving is that in the former there is no problem space, no search. The methods are almost certain of success. A skilled performance, like manuscript editing, is thus a special case of problem solving.

Of course, whether a user will exhibit problem solving or skilled performance depends not only on the task but on the state of the user as well. A nice example of the transition between one and the other can be seen in Simon and Reed's (1976) experiment on the five missionaries/five cannibals problem. The problem space for this task is

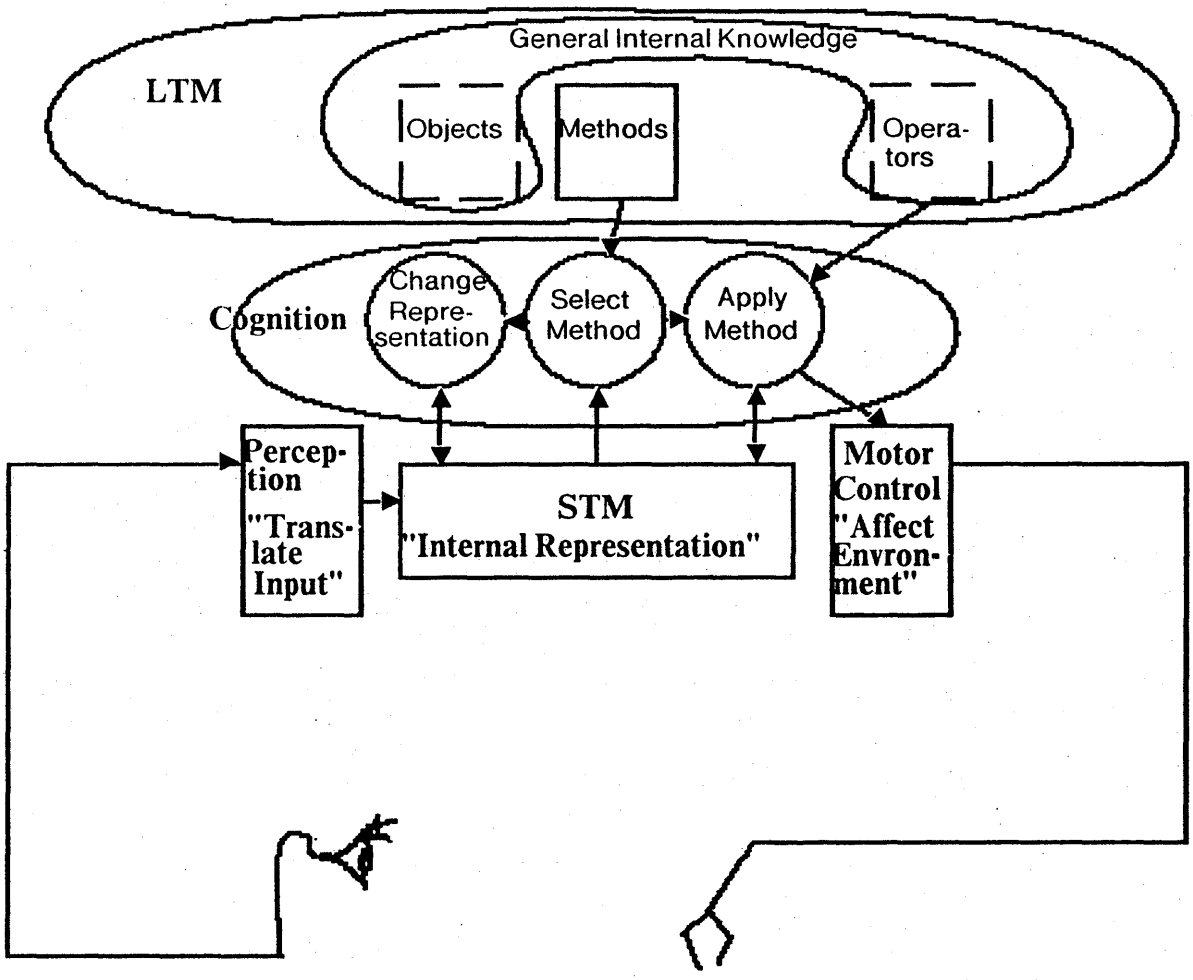


Figure 8.2. Organization of problem solver (redrawn from Figure 4.1 of Newell and Simon, 1972)

given in Figure 8.3. The subjects' searches through the problem space were characterized by two strategies: a "balance" strategy which emphasized balancing the number of missionaries and cannibals on each side of the river and a "means-ends" strategy which emphasized ferrying the maximum number of persons across the river. Subjects initially used the (defective) balance strategy and later shifted to the means-ends strategy. When subjects were retested on this problem after a successful solution, they shifted earlier to the successful strategy. In other words, as a result of practice, part of the searching behavior was eliminated and replaced by the selection of and application of methods more certain of success. Presumably after a number of tries at the problem, they would embrace the means-ends strategy from the start and follow the route marked on the figure. All search would be eliminated. While Simon and Reed's subjects exhibited problem solving behavior on their first trial, by their second attempt at the problem they had already begun to shift toward skilled performance.

Relation to other skilled behavior

Manuscript editing, as performed by the users studied in this thesis, is clearly an example of skilled performance, characterized as it is by "competent, expert, rapid, and accurate performance" (Welford,1968; p. 12). To emphasize the user's mental engagement in the task, the unruffled way in which each new problem can be paired with an old and trusty method, and the close relationship of the manuscript editing task to many other tasks of this sort in daily life, we shall speak of the manuscript editing as a *routine cognitive* skill.

Although it is not possible to state a general theory of routine cognitive skill at this point, it is possible to note some ways in which manuscript editing appears to be specialized, and thus to indicate where the general theory might be significantly different from the one presented here. Dimensions on which routine cognitive skills can vary are listed below:

(1) *Unit task structure:* Manuscript editing is structured into a sequence of almost totally independent unit tasks, each a few seconds duration. This provides an extremely short time horizon for the integration of behavior. There are routine cognitive skill tasks with even shorter unit tasks, such as making change. More often, the relevant task duration is much longer, such as cataloging a book for a library. However, it is most important to note that many routine cognitive skills such as typing (Shaffer,

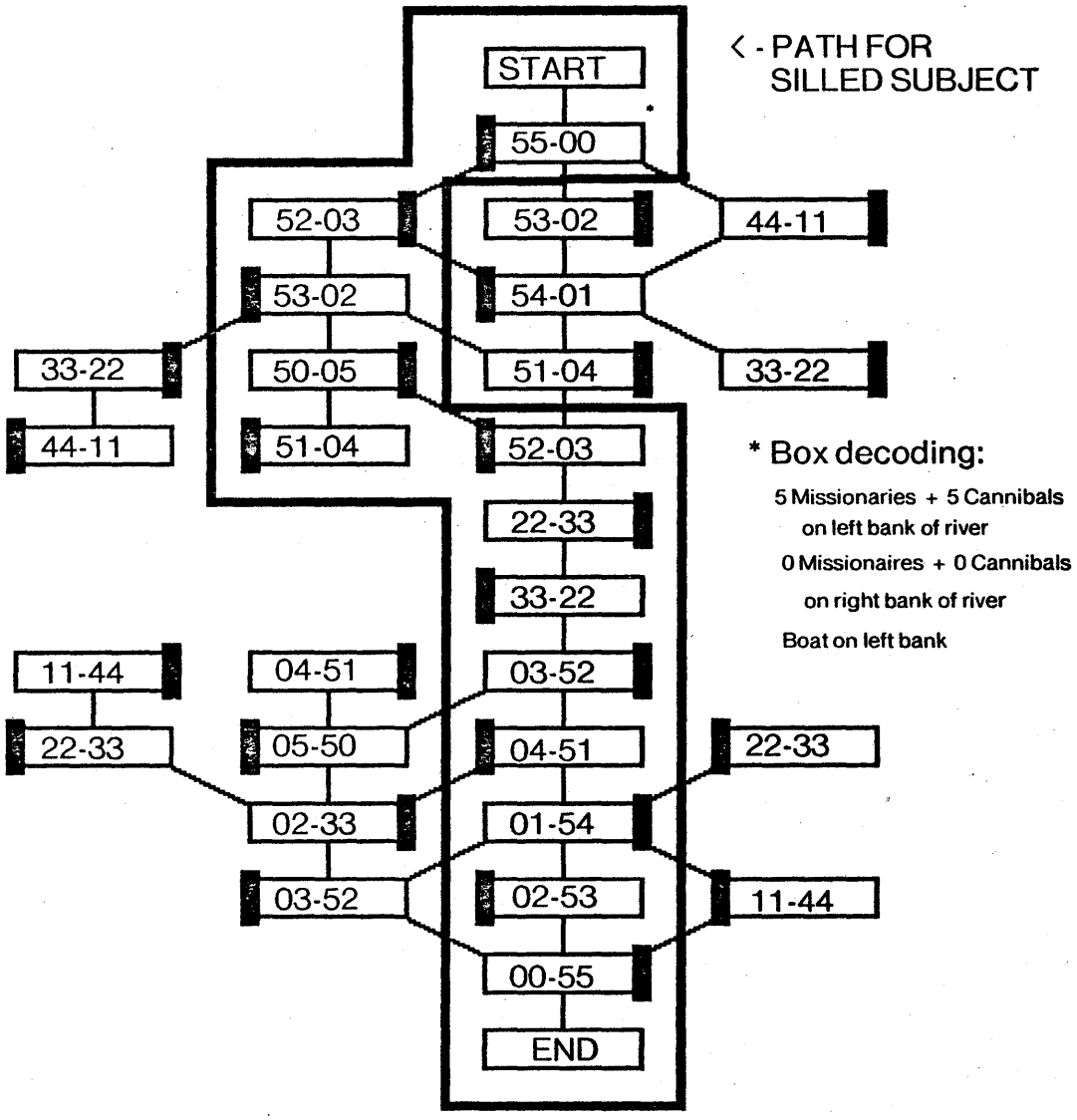


Figure 8.3. Space of legal moves for the 5-Missionaries/5-Cannibals problem (after Simon and Reed, 1976, Figure 1)

1976) do not have any unit task structure at all, but are essentially continuous activities.

(2) *Activity role of the task environment:* In manuscript editing, the task environment is not passive, as in the task of writing a business letter by hand, but responsive to the user's actions. Yet neither does task environment initiate new action. By contrast, in the task of taking airline travel reservations over the phone, the task environment is the major agent of initiation.

(3) *Payoff characteristics:* The usual trade-off between speed and accuracy is reflected in this dimension: consider the difference between typing a rough draft and typing a final copy. Both speed and accuracy are important in manuscript editing, with absolute priority given to accuracy. The emphasis on accuracy plus the detectability of errors leads to the conversion of errors to time.

(4) *Input to LTM:* With respect to knowledge in long-term memory, a distinction may be drawn between memory for the components of the skill itself, and memory for the objects in the environment being processed by the skill. It is this latter to which we refer here. In the game of bridge, the task specific knowledge players must input into LTM is fairly large, for they must be able to remember all the cards played. There is little long-term task specific information to remember in manuscript editing other than the assimilated knowledge of the skill of the specific editor.

(5) *Retrieval from LTM:* A security guard must draw on LTM frequently and rapidly to recognize many faces, whereas the information needed by an elevator operator is provided by the task environment. Modest amounts of information need to be retrieved from LTM for manuscript editing consisting largely of information about the editing system and small amounts of information about the specifics of the editing task.

(6) *Type of cognitive activity:* In manuscript editing the primary type of cognitive activity is interpretation: interpretation of the instructions of the manuscript as commands and interpretation of the feedback from the editor in terms of successful performance of the corrections to be made. There are no forms of routine reasoning, design, or evaluation activity in manuscript editing. In contrast, the electrician's task of installing a new fixture may require considerable routine inference to discover the connections for circuits only partially seen. (What differentiates the

electrician's task from problem solving is that the problems are familiar and the solution schemata previously stored, e.g., in the National Electrical Code).

(7) *Degree of motor involvement:* This dimension lies along the major discrimination of motor skills from cognitive skills. Motor involvement in many cases implies the lightening of the cognitive load, since the motor activity may not demand too much cognitive involvement. Dictating to a secretary is much more cognitively intense than composing on a typewriter, where the cognitive activity may pause periodically while the motor activity catches up. In the manuscript editing task, 40% of S13's time was spent in motor operations.

(8) *Requirement for inventing plans:* The issue here is whether a goal hierarchy (created from a limited stock of goal types) is sufficient to control behavior or whether new plans have to be constructed and then interpreted. Simple manuscript editing, such as we have been discussing, requires no planning; the goal structures provides adequate control. But if the editing task were made more complex—say, by requiring shuffling of manuscript sections—then planning would be required to decide on a correct and efficient order in which to make the changes.

(9) *Conditionality:* Conditionality is the extent to which the operators recombine in different sequences. At one extreme are the single, repetitive, assembly-line sequences studied by industrial engineers. At the other end of the scale is the short-order cook, simultaneously processing a random mixture of pancakes, eggs, and home-fries. His behavior is highly conditional, deriving from the large number of different combinations of orders that can come even from a small menu. Manuscript editing sits in the middle, having only a moderate amount of conditionality, expressed mainly in the selection of methods and in the optional choices of operators within methods.

NOTES

1 It is commonly assumed that LTM is very large, large enough so that it's physical capacity is not a limitation on performance.

2 Klatzky, 1975.

3 Identified with the time necessary to do the letter name matching in the Posner letter task (Posner, Boies, Eichelman, and Taylor, 1969).

4 Dansereau, 1968.

5 Welford, 1960.

References

Abruzzi, A. *Work, workers, and work measurement.* New York: Columbia University Press, 1956.

Birn, S. A., Crossan, R. M., and Eastwood, R. W. Measurements and control of office costs, Master clerical data. New York: McGraw-Hill, 1961.

Bolt, Beraneck, and Newman, Inc. *TENEX Text Editor and Corrector* (Manual DEC10-NGZEB-D). Cambridge, Massachusetts: Author, 1973.

Book, William Frederick. *The psychology of skill with special reference to its acquisition in typewriting.* University of Montana Studies in Psychology, 1908, **1**. Republished, New York: Gregg, 1925.

Bryan, W. L. and Harter, N. Studies in the physiology and psychology of the telegraphic language. *Psychological Review,* 1897, **4**, 27-53.

Bryan, W. L. and Harter, N. Studies on the telegraphic language. The acquisition of a hierarchy of habits. *Psychological Review,* 1899, **6**, 345-375.

Card, S. K., Moran, T. P., and Newell, A. *The manuscript editing task* (P76-00082). Palo Alto, California: Xerox Palo Alto Research Center, December, 1976.

Claude, J. A comparison of five variable weighting procedures. *Educational and Psychological Measurement,* 1972, **32**, 311-322.

Craik, K. J. W., and Vince, M. A. Psychological and physiological aspects of control mechanisms. *Ergonomics,* 1963, **6**, 419-440.

Dansereau, D. *An information processing model of mental multiplication.* Unpublished PhD dissertation, Department of Psychology,

Carnegie-Mellon University, 1968.

De Jong, J. R. The effects of increasing skill on cycle time and its consequences for time standards. *Ergonomics*, 1957, **1**, 51-60.

Deutsch, P. L. and Lampson, B. W. An online editor. *Communications of the ACM*, 1967, **10**, 793-803.

Devoe, D. B. Alternatives to handprinting in the manual entry of data. *IEEE Transactions on Human Factors in Electronics*, 1967, **HFE-8**, 21-31.

Elkind, J. I. and Sprague, L. T. Transmission of information in simple manual control systems. *IRE Transaction on Human Factors in Electronics*, 1961, **HFE-2**, 58-60.

Engelbart, D. C. and English, W. K. A research center for augmenting human intellect. *Proceedings of the 1968 FJCC*, **33**, Part 1, Montvale, New Jersey: AFIPS Press, 1968, 395-410.

English, W. K., Engelbart, D. C., and Berman, M. L. Display-selection techniques for text manipulation. *IEEE Transactions on Human Factors in Electronics*, 1967, **HFE-8**, 5-15.

Fitts, P. M. The information capacity of the human motor system in controlling amplitude of movement. *Journal of Experimental Psychology*, 1954, **47**, 381-391.

Fitts, P. M., and Peterson, J. R. Information capacity of discrete motor reponses. *Journal of Experimental Psychology*, 1964, **67**, 103-112.

Fitts, P. M., and Radford, B. Information capacity of discrete motor responses under different cognitive sets. *Journal of Experimental Psychology*, 1966, **71**, 475-482.

Glencross, D. J. Control of skilled movement. *Psychological Bulletin*, 1977, **84**, 14-29.

Goodwin, N. C. Cursor positioning on an electronic display using lightpen, lightgun, or keyboard for three basic tasks. *Human Factors*, 1975, **17**, 289-295.

Kinkead, R. Typing speed, keying rates, and optimal keyboard layouts. *Proceedings of the 19th Annual Meeting of the Human Factors Society*, 1975.

Klatzky, R. L. *Human memory: structures and processes.* San Francisco: W. H. Freeman and Company, 1975.

Knight, A. A., and Dagnall, P. R. Precision in movements. *Ergonomics,* 1967, **10**, 321-330.

Long, J. Visual feedback and skilled keying: differential effects of masking the printed copy and the keyboard. *Ergonomics,* 1976, **19**, 93-110.

Maynard, H. B. *Industrial engineering handbook, 3rd Ed,* New York: McGraw-Hill, 1971.

Maynard, H. B., Aiken, William M., and Lewis, J. F. *Practical control of office tasks with universal office controls.* Greenwich, Connecticut: Management Publishing Corp., 1960.

McCormick, E. J. *Human factors in engineering and design.* New York: McGraw-Hill, 1976.

Meister, D. *Behavioral foundations of system development.* New York: Wiley and Sons, 1976.

Mills, R. G., & Hatfield, S. A. Sequential task performance: task module relationships, reliabilities, and times. *Human Factors,* 1974, **16**, 117-128.

Newell, A. Production systems: models of control structures. In W. G. Chase (Ed.), *Visual information processing.* New York: Academic Press, 1973.

Newell, A., and Simon, H. A. *Human problem solving.* Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Oren, S. S. A mathematical theory of man-machine text editing. *IEEE Transactions on Systems, Man, and Cybernetics,* 1974, SMC-4, 256-267.

Oren, S. S. A mathematical theory of man-machine document assembly. *IEEE Transactions on Systems, Man, and Cybernetics,* 1975, SMC-5, 256-267.

Parsons, H. M. *Man-machine systems experiments.* Johns Hopkins University Press, 1972.

Payne, D. and Altman, J. W. *An index of electronic equipment operability: report of development.* American Institutes for Research, January, 1962.

Pew, Richard W.; Baron, Sheldon; Feehrer, Carl E.; and Miller, Duncan C. *Critical review and analysis of performance models applicable to man-machine systems evaluation.* BBN Report 3446. Cambridge, Massachusetts: Bolt Beranek and Newman Inc, March 1977.

Pierce, J. R., and Karlin, J. E. Reading rates and the information rate of the human channel. *Bell System Technical Journal,* 1957, **36**, 497-516.

Posner, M. I., Boies, S. J., Eichelman, W. H., and Taylor, R. L. Retention of visual and name codes of single letters. *Journal of Experimental Psychology,* 1967, **73**, 28-38.

Poulton, E. C. *Tracking Skill and Manual Control.* New York: Academic Press, 1974.

Quick, J. H. *Work factor time standards.* New York: McGraw-Hill, 1962.

Riddle, Elizabeth A. *Comparative study of various text editors and formatting systems.* Washington, D.C., Air Force DataServices Center, The Pentagon, August, 1976, (AD-A029 050).

Russell, D. S. *POET: a page oriented editor for TENEX. University of Utah, Computer Science Division,* 1973.

Savitsky, S. *Son of STOPGAP* (SAILON 50.1). Stanford, California: Stanford Artificial Intelligence Laboratory Operating Note, Stanford University, September 1969.

Seifert, D. J., Wortman, D. B., and Dukey, S. D. SAINT: A combined discrete/continuous network simulation technique. *Proceedings of the Eighth Annual Pittsburgh Modeling and Simulation Conference,* April 1977, (AMRL-TK-77-24).

Shaffer, L. H. Intention and performance. *Psychological Review,* 1976, **83**, 375-393.

Sheridan, T. B. and Ferrell, W. R. Man-machine systems: information, control, and decision models of human performance. Cambridge, Massachusetts: MIT Press, 1974.

Simon, H. A. *The sciences of the artificial.* Cambridge, Massachusetts: M.I.T. Press, 1969.

Simon, H. A. and Reed, S. K. Modeling strategy shifts in a problem-solving task. *Cognitive Psychology*, **8**, 86-97.

Stanford Center for Information Processing. *Wylbur/370--the Stanford timesharing system--reference manual, 3rd ed.* Stanford University, Novemeber 1975.

Streib, M. I. *The Human Operator Simulator: Volume I, Introduction and overview.* Analytics Technical Report 1117-1, August 1975.

Taylor, D. W. Learning telegraphic code. *Psychological Bulletin*, 1943, **40**, 461-487.

Tulloss, R. E. *The learning curve--with special reference to the progress of students in telegraphy and typewriting.* Unpublished PhD dissertation, Harvard University, 1918.

Wainer, H. Estimating coefficients in linear models: it don't make no nevermind. *Psychological Bulletin*, 1976, **83**, 213-217.

Welford, A. T. *Ageing and Human Skill.* Westport, Connecticut: Greenwood Press, 1958.

Welford, A. T. The measurement of sensory-motor performance: survey and reappraisal of twelve years' progress. *Ergonomics*, 1960, **3**, 189-230.

Welford, A. T. *Fundamentals of Skill.* London: Methuen, 1968.

Welford, A. T. *Perceptual and Motor Skills.* Glenville, Illinois: Scott Foresman and Company, 1976.

Young, R. M. *Seriation by Children: An Artificial Intelligence Analysis of a Piagetian Task.* Basel: Birkhauser, 1976.

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

 Printed in U.S.A.