Chapter 6

User Effort in the Command Specification Task

The task-artifact and behavioural studies, described in Chapters 2-5, examine two different perspectives of the general question of why users do not make greater use of history tools. Those studies have shown two possible reasons: integrated history tools do not exist to support the seven uses of history tools and designs of history-for-reuse tools do not take locality of user recurrences into account. This chapter explores a third perspective related to the user effort involved in issuing a recurrent operation. In particular, we explore the question: ‘‘Does the effort involved in using history tools to specify a recurrent operation outweigh the effort involved in simply specifying the operation from scratch?’’

Also, our study uses a cognitive modelling framework to generate models describing the human information processing operations and performance times for the typing and history-tool methods. Since these models represent the cornerstone of our analysis of user effort, it is important that their predictions be reasonably accurate. Therefore, we also examine the error in the models’ predictions.

We begin with a description of our user effort study and its findings. Then, we present an experiment we conducted to compare the predicted times against the observed times. Finally, we make some remarks regarding the cognitive modelling methodology and ideas for future work.

User Effort Study

The primary objective of this study is to compare and contrast the user effort associated with various ways of specifying recurrent commands based on an analysis of the human information processing requirements. The next few subsections describe the following aspects of the study.

- analytic methodology
- user effort
- command-specification task and methods
- constraints of the study
- estimates for predictions
- procedure
- results and discussion

Analytic Methodology

Cognitive modelling is a description and analysis of the knowledge and subtasks used to carry out a task. This type of a *task analysis* is carried out with the benefit of psychological models of human behaviour. The knowledge and task descriptions produced in the analysis are cognitive models. These models provide predictions about user behaviour (e.g., consistency, performance times).

There are four reasons for using cognitive modelling. First, it employs psychological theories of human behaviour. Hence, predictions and evaluations are grounded on these theories. Second, a goal of cognitive modelling is to strive to predict average user behaviour to within 80% accuracy while expending 20% of the effort normally required in prototyping and testing [John, 1988]. Thus, cognitive models provide designers with analytic tools to predict, prior to prototyping, how users will interact with computer systems in an approximate and reasonably accurate manner. Third, cognitive modelling provides a scientific foundation upon which we can systematically investigate and extend our investigations of user effort. Specifically, we can extend our current study within this framework to include tasks involving partial reuse and modification of recurrent commands or even command generation.

Our study uses a cognitive modelling framework consisting of the Model Human Processor (MHP for short) and Goals, Operations, Methods, and Selection Rules (GOMS, for short). MHP is a general characterization of human information processing. It consists of a system architecture and a set of quantitative parameters of component performance. The architecture is made up of three processors – perceptual, cognitive, and motor – and their associated memories, and a set of principles of operation. Each processor operates serially within itself and concurrently with all the others, subject to serial limitations imposed by data flow. Elemental acts of the processors are referred to as *operators* .

GOMS is a representation for describing the user knowledge and actions needed to perform a task. A task analysis in this framework involves subdividing a task into the gross functions performed by each MHP processor and decomposing these functions into sequences of operators. The resulting cognitive model is a description of the sequence of human information processing operators involved in carrying out a user task. One of the principal predictions made by these cognitive models is task performance times. MHP assumes that operators act independently of each other so that the time to perform a sequence of operators is the sum of the times to perform each operator.

The MHP/GOMS framework permits the formulation of GOMS models at different levels of task analysis (e.g., unit task, keystroke, and immediate behaviour levels). The Keystroke Level Model is a GOMS model formulated at the level of the MHP processes. It predicts the task performance time given the sequence of operations for performing a task. The GOMS Model of Immediate Behaviour is formulated within the MHP architecture [John, 1988]. Immediate behaviour is the direct mapping of a stimulus into a response without problem solving or planning. Such behaviours are routine and are generally associated with reaction-time tasks where the response to a stimulus is well known and initiated upon presentation of the stimulus without deliberation about what is appropriate (i.e., stimulus-response compatibility tasks – SRC for short). Our study concentrates on command specification which includes many of the routine activities bridged by the keystroke and immediate behaviour levels. Hence, our study makes use of both GOMS models.

A longer description of the MHP/GOMS framework, the task description language and the cost prediction scheme appears in Appendix C. An in-depth description of this framework, its application, and its various formulations can be found in Card, Moran, and Newell (1983) and John (1988).

User Effort

There are two user effort components: *mental* and *physical* effort. Mental effort is expended when cognitive and visual processing resources are used while physical effort is expended when motor processing resources are used.

We chose to examine user effort for three reasons. First, user effort is a factor which can influence a user's decision to use history tools. Second, we suggested in our studies of user recurrences that history tools can alleviate the physical effort in issuing a recurrent command, and a formal study of user effort would corroborate this suggestion. Third, unlike many other factors influencing history use, which are rather subjective or qualitative, user effort can be objectively examined and quantified.

Since we use the MHP/GOMS framework in our study, the user effort predictions are functions of the types and numbers of elemental MHP acts (i.e., operators) and the task performance times. The numbers and types of MHP operators are derived from a task analysis of each proposed method of command specification. Task performance time is measured from the time after users generate commands in their mind to the time they submit them to a system.

Command-Specification Task and Methods

There are three subtasks involved in issuing a command: *command generation*, *method choice*, and *command specification*. Command generation is the task of mentally generating the command to be issued. Method choice is the task of choosing the method to externalize the desired command. Command specification is the task of issuing the desired command. Our user effort analysis focuses on the command-specification task. In particular, we are interested in the user effort associated with issuing the desired command for a given method and design.

Our analysis does not deal with method choice because this decision-making task is driven primarily by subjective considerations (e.g., personal preferences, methods that come immediately to mind, user's awareness of the available methods) and because cognitive models do not address such considerations. Therefore, we opted to study user effort in an objective manner (i.e., given a method and design, determine the user effort).

There are four reasons for ignoring a command-generation task. First, command generation involves plan generation and problem solving. which are beyond the scope of current cognitive modelling efforts. Second, the GOMS models used in our analysis assume that user behaviours are routine and involve no planning or problem solving. Third, command generation is poorly understood in HCI (i.e., is command specification intertwined with or separate from command generation?). Finally, a study of command generation constitutes several theses in itself.

Because our understanding of the command generation task is limited, we made two simplifying assumptions for our study. First, we assume that command generation and command specification are independent. Second, we assume that users have generated the full command prior to issuing it and this command is available in working memory. These assumptions permit an initial investigation of the issue of user effort involved in issuing a command. However, they may be a gross simplification of what happens typically when users issue commands. Hence, these assumptions represent a limitation of our study.

Two methods are available for specifying a recurrent command: typing and using a history tool. Three history-tool designs involving recall and/or recognition capabilities are examined. In the *command feature* class, a user *recalls* a feature of the desired command to be invoked; in the *history menu* class, a user *recognizes* the desired command from a visual display of the history; and in the *step buffer* class, a user uses one or a combination of recall and recognition capabilities to peruse and select individual history items.

Command Feature

This class of designs supports the recall of a command from a user history based on a feature that the user provides to uniquely identify it. The four proposed designs are based on different capabilities supported by *csh*.

Name – *!*name repeats the most recent command with the given *name*.

Number – *!number* repeats the command with the given *number*.

Argument – *!?argument* repeats the most recent command with the *argument*.

Last Command – *!!* repeats the most recent command.

History Menu

This class of designs presents a portion of a user history in a menu whose items are selected using a mouse. Two variations are considered:

Popup – A user brings up the menu and scans and selects the desired command.

Static – A user scans a static menu and selects the desired command.

Step Buffer

This class of designs allows the user to single step through a user history from the most recent to the least recent to locate the desired command. Two of the designs are directly supported in *tcsh* while the third is based on a related feature available in *csh* and *tcsh* – file name completion:

Single Step – A user single steps through the history to the desired command.

Recall & Step – A user recalls how recent the command is and single steps to it.

History completion on command name – A user types a command name and presses the *history completion key*. The system searches the history for the most recent command matching the given name and presents it to the user. If the matching item is not the desired one, the user presses the *history completion key* again and the system finds the next most recent command matching the given name. This is repeated until the desired command is presented or no more commands match.

Constraints of the Study

Our primary research objective is to explore the issue of user effort in the specification of a recurrent command using cognitive modelling as the investigative methodology and not the development of the methodology. Thus, our study is subject to the limitations that exist with the cognitive modelling approach.

One consequence of these limitations is that we are unable to model the task of command generation or the task of deciding which method and design to use for command specification because these tasks are not modelled by the methodology. Despite this, there is value in studying the command specification task independent of the command-generation and decision-making concerns. In particular, we can explore the advantages and disadvantages of methods involving the use of history tools and typing.

A second consequence of these limitations is that we only model expert, error-free, user behaviour associated with command specification. However, there is value in such a study because we can examine the issue of user effort under optimal conditions.

A third consequence of these limitations is that parallel processes are modelled either as a series of processes or as a single process (i.e. the one requiring the longest time to completion); the serial approach is simpler to implement but leads to conservative predictions of task performance times while the one-parallel-process approach is difficult to implement but more accurate. The one-parallel-process approach is based on John (1988)'s proposal of using the critical path technique to model parallel processes. The technique involves specifying the component processes, their duration, and their inter-dependencies to form a network and then determining the path through this network that takes the longest time. We opted to use this critical path approach but we did not explicitly formulate the network in the course of generating our models; we simply selected the process from the group of parallel processes that was considered to be the critical one. Our informal use of the critical analysis approach (i.e., do not detail insignificant parallel processes) is acceptable (but perhaps not as thorough in the documentation of these parallel processes) as long as the predictions are reasonably accurate and we adhere to two important criteria of cognitive modelling. These two criteria are: a) adequate coverage of the total task (i.e., take into account all first-order effects observed in the task domain) and b) approximation of the underlying processes (i.e., cognitive models, as an engineering prediction tool, need only include sufficient detail to do the design job) [Card, Moran, & Newell, 1983; John, 1988].

Aside from our study's constraints, there is also the concern regarding our claim that history-based, command-specification is a routine cognitive skill. Three arguments are made in support of this claim. First, a characteristic of skilled routine performance is the absence of learning. This occurs when task performance improves, as a result of learning, over many repeated encounters with a task until it reaches a plateau representing skilled behaviour. The highly repetitive nature of user interactions (recall the observed recurrence rate of 75%) suggests that there are plenty of opportunities for issuing recurrent commands and using history tools. Hence, users can become skilled with history-based command specification. Second, another characteristic of skilled routine performance is the routine nature of a task. That is, people generally become skilled in whatever becomes routine for them [Card, Moran, & Newell, 1983]. Since the subtasks (e.g., menu selection, search of a menu with a known organization, keystroke, recognition of a target item) associated with history-based command specification are prevalent in many of today's user interfaces, they are a routine part of user interactions and users can become skilled with such subtasks. Finally, the frequency with which user interactions are repeated suggests that history usage can also become routine. Therefore, in the context of our particular command specification task, history use can be considered a routine cognitive skill.

Estim ates for Predictions

Our study uses a number of *MHP operators* and *application-specific parameters*. Each MHP operator is a specific elemental act of one of the three MHP processors (e.g., M_K is the keystroke operator associated with the motor processor M). Each application-specific parameter characterizes an aspect of the UNIX command specification task (e.g., $\bar{n}_0$ is the average number of characters in a UNIX command name). The following subsections enumerate these operators and parameters.

MHP Operators

Table 1 lists the durations of the various MHP operators used in our analysis. Three of the MHP operators were not available in Card, Moran, and Newell (1983)'s original list and had to be obtained elsewhere: menu search, associative memory retrieval, and menu selection with a mouse.

Menu Search. We estimated the menu search operator from an experiment (see Appendix D) because there were no prior estimates. The experiment examined the task of visually searching a menu ordered from most recent to least recent references of the menu's items. Our empirical data suggest that the time to search such a menu is approximated by a linear function of the position of the target item in the menu. We used the mean intercept and slope estimates obtained for the locality conditions (see Table 5 in Appendix D). The reason for focusing on

Parameters for Component Processes		Time (msec)	Source
Cognitive Operators			
C_E	Execute a mental step	70	OO(240)
C_M	Retrieve from memory	1200	John(44)
C_{CRT}	Choice reaction time	340	CMN(74)
Perceptual Operators			
P_I	Perceive an image	100	CMN(32)
P_E	Eye movement	230	CMN(25-28)
P_{S_0}	Menu Search – Intercept	690	Appendix D
P_{S_1}	Menu Search – Slope	110	Appendix D
$P_S(l)$	Menu Search – Combined	$690 + 110\ l$	Appendix D
Motor Operators			
M_B	Press a button	70	CMN(69)
M_K	Type a character		
	Best typist (135 wpm)	80	CMN(264)
	Good typist (90 wpm)	120	CMN(264)
	Average skilled typist (55 wpm)	200	CMN(264)
	Average non-secty typist (40 wpm)	280	CMN(264)
	Worst typist	1200	CMN(264)
$M_{P_M}(l)$	Point to the lth item in menu	$-107 + 223 \log_2(l+1)$	MSB(165)
$M_{D_M}(l)$	Drag to the lth item in menu	$135 + 249 \log_2(l+1)$	MSB(165)
M_H	Home to mouse or keyboard	360	CMN(237)

CMN(32)	[Card, Moran, & Newell, 1983] on page 32.
Appendix D	See Table 5, Appendix D.
John(44)	[John, 1988] on page 44.
MSB(165)	[MacKenzie, Sellen, & Buxton, 1991] on page 165
OO(240)	[Olson & Olson, 1990] on page 240.

Table 1: MHP Operators used in the user-effort analysis.

locality rather than non-locality behaviour is that our studies in Chapter 5 suggest that history tools for reuse are extremely useful during periods where clusters of user command line references recur (i.e., phases) as opposed to periods in which user command line references are shifting from one locality set to another (i.e., transitions).

Associative Memory Retrieval. A recent extension to the MHP/GOMS framework, prediction of user behaviour in command-abbreviation tasks, provided an estimate for the cost of retrieving a completely arbitrary association between a stimulus word and its required letter-combination [John, 1988]. Because several of our history-based designs involve an associative form of memory retrieval (e.g., actions for the history task, a number for a recurrent command, an argument from a recurrent command), we assume that this estimate is appropriate for our purposes. However, our memory retrievals are not necessarily arbitrary associations.

Thus, John (1988)'s estimate may be somewhat conservative for our study (i.e., it provides an upper bound). The appropriateness of using this estimate is examined later in an experiment.

Menu Selection with a Mouse. Fitts (1954) demonstrated that the time to move a distance A to acquire an object of size W varies with A and W:

$$MT = a + b \log_2 (2A / W) \qquad \text{Eqn. (1).}$$

Recently, MacKenzie (1989) noted that when A is small and W is large, the *log* term in Eqn. (1) is negative and that this is theoretically unsound. Instead, MacKenzie (1989) proposed the following corrected formulation of Fitts' Law:

$$MT = a + b \log_2 (A / W + 1) \qquad \text{Eqn. (2).}$$

Using this formulation, MacKenzie, Sellen, and Buxton (1991) empirically determined values for a and b for point and drag movement tasks involving a mouse. These results are used as estimates for our two kinds of mouse selection operators (see Table 1). The "point with mouse" operator involves moving a mouse to an object and selecting it (e.g., static menu selection) while the "drag with mouse" involves moving a mouse to an object with the mouse button held down and selecting it by releasing the button (e.g., popup menu selection).

We assume that the mouse is always located W units above the menu and that the height of each menu item measures W units. Therefore, to move to the lth item in the menu, the distance travelled is lW and the resulting log term is replaced by $\log_2 (l+1)$.

Application-Specific Parameters

Table 2 lists the values for parameters characterizing UNIX command specification and history use. They include: probability of a command line having no argument, average number of characters in a command name and command argument, and average number of arguments. The values for all but two parameters are derived from an analysis of Greenberg (1988a)'s UNIX command traces. We elaborate on our choice of the values for the two exceptions.

The value for the probability of locating a command in each menu position (p_l) is chosen to simulate the locality behaviour and is identical to the one used in our menu search experiment. We did not use the same probability distribution observed in our studies of locality in Chapter 5. The reason is that the full range of locality set sizes (observed for real user sessions of 500 or more user commands) cannot be recreated in our two experiments (i.e., menu search and accuracy) in which user sessions were only 100 commands long. In addition to scaling down the length of a typical user session for the purposes of our experiment, we scaled down the locality set sizes that occur and we used a probability distribution that reflects the observed trend in the distribution of these 8 locality set sizes (i.e., a large number of locality sets of size 1 to 4). For the same reason as that presented for the menu search operator (see Appendix D), our study focuses on locality rather than non-locality behaviour.

The value for the average number of command lines with the same command name in a locality set (e) is derived as follows. Our analysis of Greenberg (1988a)'s UNIX user traces indicates that no more than three command lines beginning with

Parameters for Characteristics of UNIX Command Specification & History Usage		Values Mean ± SD
Probability of Command Line		
p_a	With no arguments	.37 ± .19
Probability of search item at menu position l=1 to 8		
p_l	Small locality sets	.26,.24,.22,.20,.02,.02,.02,.02
Average Number of		
e	Similar command lines examined	2
a	Command arguments	1.56 ± .52
Average number of characters in		
$\bar{n}_0$	Command name	3.28 ± .57
$\bar{n}_A$	Command arguments	5.31 ± 1.37
Average Number of Menu Items Examined		
$\sum_{l=1}^{8} lp_l$	Small locality set	2.72

Table 2: Table of estimates related to characteristics of UNIX command specification and UNIX history usage required for our current study.

the same command name occur within the same locality set. We did not examine the user traces to derive the exact probability distribution because it would involve a great deal of work. Instead, we assumed that all three cases occur with equal probability. Therefore, e is the expected value of observing 1, 2, and 3 occurrences of a command line with the same command name in a locality set (i.e., $e = E(s) = \sum_{s=1}^{3} sp(s) = 2$).

Procedure

Using MHP as a characterization of human information processing, we generate a description of the MHP operations involved in the command-specification task for each proposed design (i.e., we generate a cognitive model). The resulting task description appears in Appendix E. The user effort associated with each proposed design is the sum of the operations in the associated cognitive model (see Appendix E). Table 3 contains the user effort equations obtained by substituting the values for the application-specific parameters.

Our analysis assumes that typing is the default course of action to take for command specification. Therefore, whenever a history tool is used, the user must switch from the default method (i.e., typing) to the history-based method. In all the history-based designs, this switching operation is manifested as an additional mental operation to retrieve the actions for the design from long-term memory. This memory retrieval requires a longer cognitive cycle C_M and represents the penalty for switching to a history-based method from the default typing method. Our assumption of a default method (i.e., typing) and thus a penalty for using the

Proposed Designs	Mental Effort	Physical Effort	System Effort
Typing	$17.5\,C_E$	$10.5\,M_K$	
Command Feature			
!!	$2C_M + 3C_E$	$3M_K$	
!name	$C_M + 7.3\,C_E$	$5.3\,M_K$	
!number	$2C_M + 5C_E$	$5M_K$	
!?argument	$2C_M + 9.3\,C_E$	$8.3\,M_K$	
History menu			
Popup	$C_M + 3C_E + \sum_{l=1}^{8} p_l P_S(l)$	$2M_H + \sum_{l=1}^{8} p_l M_{D_M}(l)$	S
Static	$C_M + 3C_E + \sum_{l=1}^{8} p_l P_S(l) + P_E$	$2M_H + \sum_{l=1}^{8} p_l M_{P_M}(l)$	
Step Buffer			
Single step	$C_M + 4.7\,C_E + P_{S_0} + 2.7\,P_{S_1} + 2.7\,P_I$	$3.7\,M_K$	$2.7\,S$
Recall & step	$2C_M + 5.7\,C_E + P_{S_0} + P_{S_1} + P_I$	$3.7\,M_K$	S
Command name with history completion	$C_M + 9.3\,C_E + 2P_{S_1} + 2P_I$	$6.3\,M_K$	$2S$

Table 3: The user effort associated with each command-specification design after substituting for the values for the application-specific parameters. The user effort is partitioned into mental, physical, and system effort components. Note, in order to simplify the notation, we use S to represent system response time. However, the value for S may be different for different designs because they may take different amounts of system time to perform.

history-based method is reasonable because, instead of providing the command directly, users are performing an additional task (i.e., using an abstraction vehicle like history tools to produce the command).

The primary focus of our analysis is to examine the mental and physical effort involved in each design proposal. To facilitate this analysis, we partitioned the user effort associated with each design into the mental and physical components as well as a system effort component (see Table 3). The mental effort component contains all cognitive and perceptual operators for a design. The physical effort component contains all simple motor, menu selection, and keystroke operators for a design. The system effort component contains all user pauses, in a design, for system response S. Note, in order to simplify the notation, we use S to represent system response time in all the designs. However, the value for S may be different for different designs because they may take different amounts of system time to perform. In our analysis, we make sparing reference to the system effort component and in most cases, we ignore this component.

Cognitive models are primarily tools for predicting user behaviour but they are also useful as ‘‘tools for thought’’ [Newell & Card, 1985]. That is, MHP/GOMS embodies a theory of human information processing and in the course of doing the cognitive modelling, user interface designers are made aware of the human information processing requirements of a particular design in the form of the number and variety of MHP operators involved. However, cognitive models as tools for thought have been underrated especially in light of the fact that current cognitive models and their predictions have many limitations and are subject to further improvements. Because current cognitive models are still in need of further development, it is premature to place most of the attention on the predictions derived from cognitive models (e.g., Gray, John, Stuart, Lawrence, and Atwood (1990) and MacLeod and Tillson (1990)) and to place little attention on the insights derived from cognitive modelling. Thus, as part of our findings, we also present the insights derived from our analysis. Furthermore, we use the task performance times and MHP estimates to aid in the comparison of designs. In particular, numerical estimates (see Table 4) are used to quantify the user effort required in each design.

Proposed Designs	Predicted Times (msecs.) Mental Effort	Physical Effort	System Effort
Typing	1220	$10.5\,M_K$	
Command Feature			
!name	1710	$5.3\,M_K$	
!number	2750	$5M_K$ [1]	
!?argument	3050	$8.3\,M_K$	
!!	2610	$3M_K$	
History Menu			
Popup	2399	720 + 577	S
Static	2629	720 + 289	
Step Buffer			
Single step	2790	$3.7\,M_K$	$2.7\,S$
Recall & step	3700	$3.7\,M_K$	S
Command name with history completion	2270	$6.3\,M_K$	$2S$

Table 4: The user effort equations in Table 3 after substituting for the all MHP operators except M_K. Note, unlike the other designs in the command feature category, the *!name* design does not require any mental retrievals including the mental retrieval of the command name because the command name is already in working memory. Note also, in order to simplify the notation, we use S to represent system response time in all the designs. However, the value for S may be different for different designs because they may take different amounts of system time to perform.

Results and Discussion

Our comparative analysis of command specification designs is divided into three parts. The first part compares the user effort associated with each history-based design class: *command feature*, *step buffer*, and *history menu*. The second part compares user effort across the three history-based design classes by examining the advantages and disadvantages of using recognition versus recall versus a combination of recognition and recall. The third part compares the use of *typing* against the use of history tools for specifying a recurrent command.

Human Performance within a Class of History Designs

Command Feature Designs. While design *!!* is less powerful than design *!number*, they are comparable in terms of the mental and physical operations that are involved (see Table 3). They both require two memory retrievals: one for the method and the other for the command number or an association between the generated command and the previous command. However, design *!number* [1] takes more time with increasing number of digits in a command number because of additional cognitive cycles C_E and keystroke operators M_K. Because the predicted times for the two designs are comparable and design *!!* takes slightly less time, the balance of the analysis uses design *!!* to represent both designs.

Designs *!name* and *!?argument* represent two different ways of recalling a recurrent command: by name or by argument. There are two reasons why *!?argument* takes more time than design *!name*. First, UNIX command arguments are longer than command names (compare $\bar{n}_A$ and $\bar{n}_0$ in Table 2) and this results in additional cognitive cycles C_E and keystroke operators M_K. Second, an extra memory retrieval C_M is required to choose a command argument which uniquely retrieves the command. Such a retrieval is not required in design *!name* as the name is available in working memory. The additional cognitive complexity in design *!?argument* may explain why only 2 of the 5 subjects in our exploratory study made use of the *!?argument* feature while all 5 subjects used the *!name* feature (see Table 2a in Chapter 4).

History Menu Designs. The *popup menu* design requires system response and drag-with-mouse operators while the *static menu* design requires eye-movement and point-with-mouse operators (see Table 3). The *static menu* design requires slightly more mental effort because of eye movement (see Table 4).

While both our mouse selection tasks involve moving a mouse to acquire an object, the tasks have different task difficulty levels [2]. MacKenzie, Sellen, and Buxton (1991) report an index of difficulty of .25 secs/bit. for the drag-with-mouse operation as opposed to .22 secs/bit for the point-with-mouse operation. Also, for an 8-item menu, a drag-with-mouse operator takes, on average, twice as long as a

[1] In any intensive user session (i.e., more than 99 commands), a command number should, on average, contain 3 digits. When a command number has only one digit, the two designs are identical.

[2] The *log* term in Eqn (2) is known as the index of difficulty and carries the units ‘‘bits’’ (because the base is ‘‘2’’). The reciprocal of the coefficient *b* is the index of performance which is the human rate of information processing for the movement task.

point-with-mouse operator (see the second value in physical effort in Table 4). We conjecture that the drag-with-mouse operation is more difficult compared to the point-with-mouse operation because of the tension involved in holding down the mouse button as the mouse is moved. Without even considering the S operator in the *popup menu* design, the time to perform the *static menu* design is slightly faster than the *popup menu* design (see Table 4).

Other design considerations point favourably to the *static menu* design. In particular, it provides continuous user feedback of a history's contents because it remains on a user's screen while *popup menu* users must bring up a menu to glance at its contents. Furthermore, a drag-with-mouse operation in the *popup menu* design has more potential for user error due to failure to hold the mouse button down while making a selection. On the other hand, a *static menu* does incur screen real estate costs but such costs can be minimal because a typical locality-set-based history menu contains one to five items (see Table 1 in Chapter 5). However, the menu may be obscured by other overlapping windows and users need to perform extra operations to unveil the menu. Nevertheless, the advantages of feedback, less errors, and less effort outweigh these disadvantages. Therefore, the balance of our analysis will use the *static menu* as the prototypical *history menu* design.

Step Buffer Designs. In all three *step buffer* designs, a user must pause after each stepping operation to wait for the system to respond (S) with the retrieved command (see Table 4); the number of pauses is proportional to the average number of items examined prior to a selection. There are more pauses in the *single step* and *command name with history completion* designs than in the *recall & step* design.

On the other hand, in the *recall & step* design, a user must accurately recall the position of a desired command and single step directly to it. The extra mental retrieval operation in this design results in more mental effort (3.7 secs.) compared to the other two step buffer designs (2.3 secs. and 2.8 secs.).

The *command name with history completion* design requires more physical effort than the other two designs because of the additional keystrokes required to type a command name. However, it requires less mental effort compared to the other two designs. In general, minimizing the use of limited cognitive resources and thus mental effort is important as it ensures that these resources are available for other tasks. Assuming average or better typing skills (M_K <= 200 msecs.), the *command name with history completion* design is a faster design.

Human Performance between Classes of History Designs

Pure Recall Designs. All *command feature* designs and the *recall & step* design are pure recall designs; they exploit a user's ability to recall characteristics of recurrent commands. Design *recall & step* requires more mental effort (see Tables 3 & 4) than all three *command feature* designs because of the perceptual elements of the design (e.g., confirmation of a system retrieval prior to selection). However, design *!?argument* requires more physical effort than any of the other pure recall designs because it requires more keystroke operators (8.3 see Table 4). Note, an alternative to the *!?argument* design would be to choose a substring to identify the *argument* which would require fewer keystrokes but it would be at the expense of more mental effort associated with determining the appropriate substring (i.e., an extra C_M operator); the additional mental effort with the *!?substring_of_argument* design would be more than the savings in physical effort. Because of the considerable physical and mental effort required in the *!?argument* and *recall & step* designs, we ignore them in the subsequent analyses.

Pure Recognition Designs. All *history menu* designs and the *single step* design are pure recognition designs; they exploit a user's perceptual/recognition abilities by providing a visual interface to a user history. The *static menu* design requires less mental effort than the *single step* design (2.6 secs compared to 2.8 secs.). If users are average or better typists (M_K <= 200 msecs.), the *single step* design requires less physical effort compared to the *static menu* design (1.0 secs. compared to .7 secs) because single-step cursor operations are faster. However, users with poor typing skills (M_K >= 280 msecs.) will be faster with the *static menu* design rather than the *single step* design because the homing-to-device and menu selection operations are faster. Also the *single step* design requires system response time, once for each item examined, while none are required for the *static menu* design. This is not an issue on fast systems.

Pure Recall versus Pure Recognition Designs. The *!name* recall design requires less mental effort than either of the two recognition designs because its associated retrieval operation involves a working-memory retrieval C_E which is simpler than the visual search operations $P_S(l)$ in the two recognition designs (see Tables 1 & 3). However, the three recall designs – *!!*, *!?argument* and *recall & step* – require more mental effort than the two recognition designs because they involve long-term memory retrieval C_M (see Tables 1 & 3).

All pure recall designs and the *single step* recognition design require typing skills (see Table 4) while the history menu designs require skill with a mouse. The physical effort in the *static menu* design (1 sec.) is more than the physical effort in the recall designs when users are better than average typists (M_K < 200 msecs.). However, if typing skills are poor (M_K >= 350 msecs), the *static menu* design requires less physical effort than all the recall designs.

Assuming that system response time is negligible, $S = 0$, Figures 1 and 2 plot the time to perform each proposed design as a function of keystroke operator times M_K; Figure 2 is Figure 1 extended to include a worst case typist (M_K = 1200 msecs). From Figure 2, we observe that the *static menu* design is superior to all recall designs when a user's typing speed is poor (M_K >= 400 msecs).

A Combined Recall and Recognition Design. The *command name with history completion* design is a combined recall and recognition design; it exploits recall as well as perceptual and recognition abilities. Users must provide the command name as a selection pattern and must inspect the retrieved command to accept or reject it. While the memory retrieval time is small, because the command name is available in working memory, this design requires perceptual, cognitive, keystroke and system response operators. In general, this combined recall and recognition design provides no substantial benefits compared to the pure recall or pure recognition designs.

Compared to the pure recall designs, this design is neither superior in cognitive (2.3 secs.) nor physical effort (6.3 M_K). In fact, the *!name* design and this design are similar with respect to retrieval of a command name but the *!name* design does not require the extra perceptual, cognitive, motor, and system response

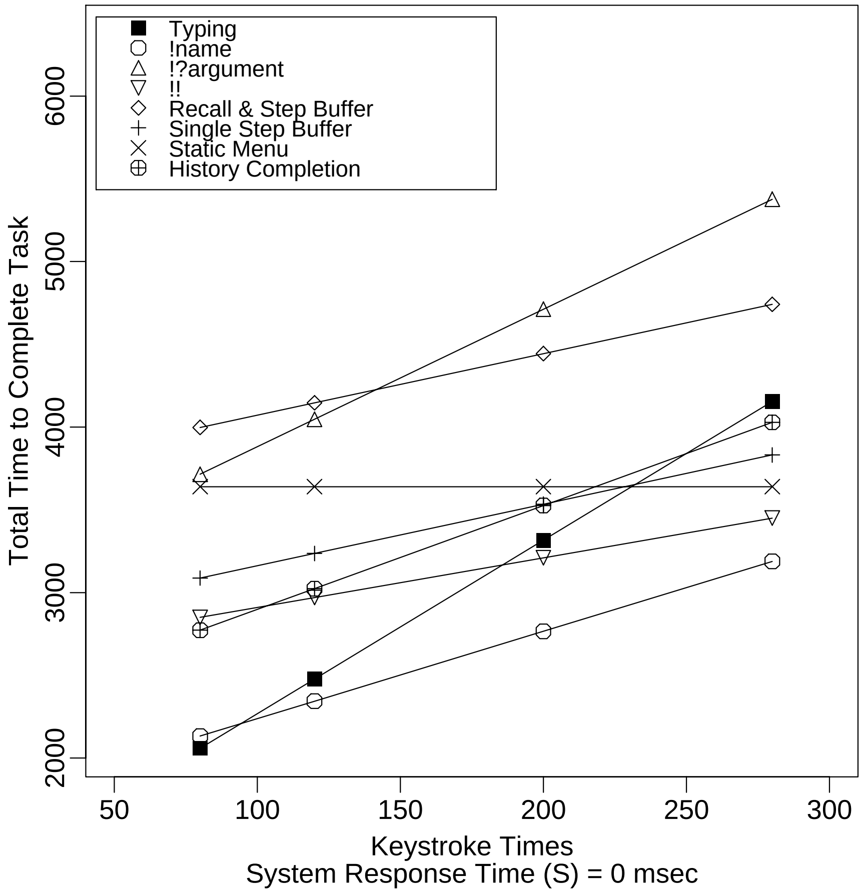

Figure 1: The time to perform each design is plotted as a function of keystroke times (M_K = 80, 120, 200, and 280 msecs.) The pure recall and pure recognition history designs are denoted by unfilled graphical markers and crosses, respectively. The combined recognition and recall history design is denoted by an unfilled graphical marker with a cross in the center. The *typing* design is denoted by the black box.

operations (see Table 3). When a user's typing skill is poor, all pure recall designs, except for *!?argument*, require less time than this design.

Compared to the two pure recognition designs, this design requires less mental effort. However, users with average to worse typing skills (M_K >= 200 msecs.) require more physical effort to use this design (see Figure 2). If system response time (S > 0) is considered, the performance time for this design will be higher.

Typing versus Using a History Tool

Typing requires the least mental effort compared to any history method (see Table 4) because we assume that the command is already in working memory and

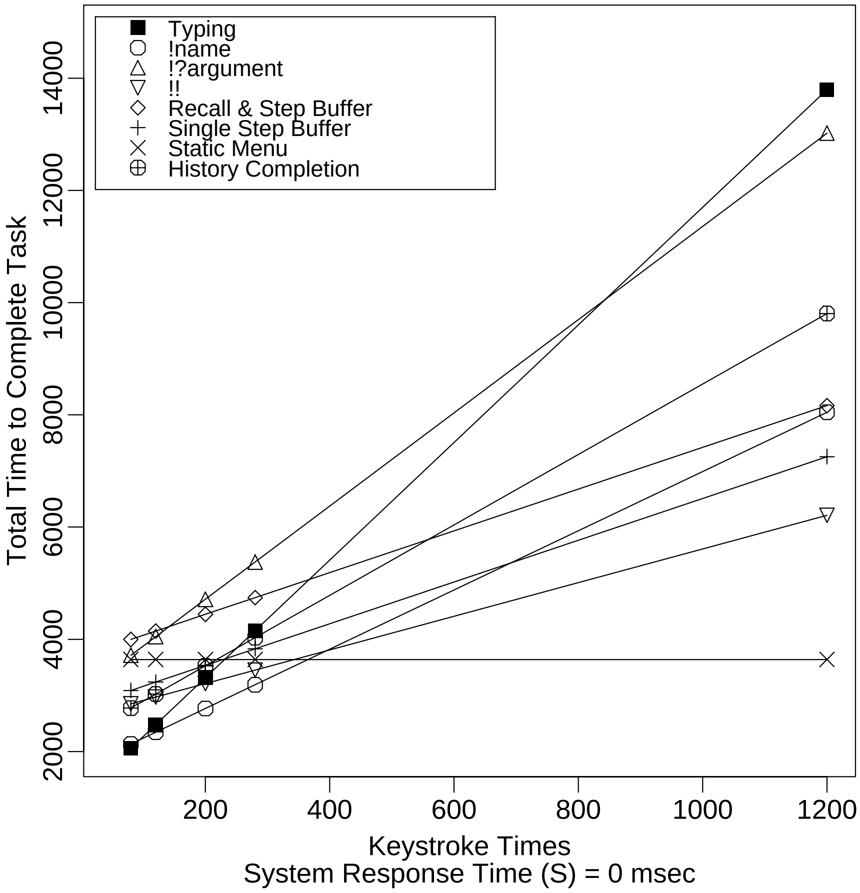

Figure 2: Figure 1 extended to include a non-typist keystroke time (i.e., M_K= 1200 msecs.). The time to perform each design is plotted as a function of keystroke times (M_K = 80, 120, 200, 280, and 1200 msecs.) The pure recall and pure recognition history designs are denoted by unfilled graphical markers and crosses, respectively. The combined recognition and recall history design is denoted by an unfilled graphical marker with a cross in the center. The *typing* design is denoted by the black box.

neither long-term memory retrieval nor visual search are involved. There are only cognitive operations associated with the selection of command line words from working memory and the initiation of motor operations for keystrokes M_K. As well, *typing* requires no point-and-select and no system response times. However, it requires the largest number of keystroke operators (10.5 – see Table 4). As a result, this method is the worst for issuing a recurrent command when non-typists (M_K = 1200 msecs.) are involved (see Figure 2). If users are expert typists (M_K = 80 msecs.), this method is the fastest. However, in the case of non-expert typists, there is always at least one history-based design faster than *typing* despite the fact that all history-based designs require a memory retrieval for the history method (C_M = 1200 msecs.).

The *!name* design is faster than *typing* and all other history designs when users have good to average non-secretarial typing skills (120 msecs. (i.e., 90 wpm) <= M_K <= 280 msecs. (i.e., 40 wpm) – see Figure 2). This occurs because it requires the least mental effort (1.7 secs.) of all history-based designs and requires half the physical effort of *typing* (5.3 M_K compared to 10.5 M_K).

The fastest recognition design, *static menu*, is the third fastest history design for users with average non-secretarial typing skills (M_K = 280 msecs. (i.e., 40 wpm)) and is the fastest history design for users with worse-than-average, non-secretarial, typing skills (M_K > 280 msecs. (i.e., 40 wpm) – see Figure 2). Furthermore, the combined recognition and recall design, *command name with history completion*, is faster than *typing* for users with worse-than-average, non-secretarial, typing skills (M_K > 280 msecs. (i.e., 40 wpm) – see Figure 2).

Accuracy of the Cognitive Models' Predictions

An important concern with modelling user behaviour (i.e., typing commands and using history tools) in a different task domain is whether the two GOMS models support the underlying activities and whether the cognitive model predictions are reasonably accurate. While some of the activities considered in our study are those used in the development of the two GOMS models (Keystroke Level Model and GOMS Model of Immediate Behaviour), there is a new activity (i.e., menu search) whose estimate had to be obtained.

Also, in the interest of keeping both GOMS models simple, the details of mental operations like long-term memory retrieval [John, 1988] and ‘‘mentally preparing’’ to execute physical operators [Card, Moran, & Newell, 1983] were not elaborated. A mental operator is used to represent such operations. As described earlier, we assume that the time parameter of such operations are estimated by the mental retrieval operator that John (1988) determined. However, it is unclear whether this is a reasonably accurate estimate of the time to perform the retrieval activities that are part of the history-tool method.

In order to resolve the two accuracy concerns (i.e., being able to model and predict history-tool activities and to use the current estimate of mental retrieval time for our study), we selected a subset of the designs examined in our study and ran an experiment to collect actual task performance times. Then we compared the cognitive models' predictions against that of the observed performance times to assess the predictive accuracy of the models.

Method

The four selected designs are the *typing* method and one design from each of the three history-tool classes: *recall by argument*, *popup menu*, and *single step buffer*. All subjects performed all four tasks a number of times with the tasks being administered in a random order. The reason for combining the trials of all four tasks instead of blocking the trials by task is to mimic a typical user session in which the particular task used to issue a recurrent command is not known *a priori*.

Subjects

18 paid students and staff participated in the experiment. All subjects had knowledge and experience with UNIX, had previous experience using a mouse, and had varying typing skills ranging from touch typists to two-finger typists.

Materials and Apparatus

A nine-item history list was used. Items were organized from the most recent to the least recent. In the popup-menu task, the eight most recently referenced items ere visible in the menu. The most recently referenced item appeared at the top of the menu and the 8th most recently referenced item appeared at the bottom of the menu. The stimuli were UNIX command lines and are listed in Table 5. For reasons similar to those in the menu search experiment, menu search features like item length and distinguishing characters (e.g., #) were either neutralized or absent.

The experiment made use of a SUN 3/50 workstation consisting of a high-resolution graphic display with a mouse and keyboard. A simple window manager – MGR – from Bellcore was used to program the experiment. The SUN display was divided into three windows. The top window contained instructions to the subjects. The bottom window displayed an error message whenever a subject made a mistake (e.g., selecting an inappropriate key on the keyboard). The middle window was the primary interaction window in the experiment. A stimulus was presented to a subject in this window and a subject's response was cued and initiated through this window.

Tasks

The experimental task is a *test* task consisting of two subtasks. Subjects were presented with a UNIX command line whose words were jumbled and they

Practice 0	Practice 1	Real Test
`tail -5 other`	`rm -i bkup/*`	`lpr -Plw *.c &`
`head -5 Msgs`	`cp bkup/* ../`	`dbx main core`
`cat m* > Msgs`	`mv mbox bkup`	`mv a.out main`
`cat msg1 msg2`	`ls -alt mbox`	`cc -c main.c &`
`tail -f save`	`pwd; du -s .`	`co -r4 main.c`
`ls -l > save`	`rm -rf srcdir`	`lint -h test.c`
`mv msg4 save`	`ls -lt srcdir`	`co -r2 test.c`
`mv msg0* dir`	`du srcdir | p`	`rlog -l test.c`
`ls -a dir | p`	`alias p more`	`make -f clean`

Table 5: The menu items used in the validation experiment.

had to unjumble it to form a legal UNIX command line [3]. This *unjumbling* task simulates command generation with the result that the unjumbled command is cached in a subject's working memory. After unjumbling the command, subjects pressed the ''carriage return'' key. Then, subjects were cued to perform one of the four command specification tasks: TYPE, PATTERN, STEP, or MENU <please use the mouse>. The TYPE cue refers to the *typing task* which involved typing out the whole generated command line followed by a ''carriage return''. In the other three cued tasks, subjects retrieved the generated command from a history using the cued history task.

The *recall-by-argument* task involved mentally selecting any word other than the first word of the generated command line and using it as a retrieval pattern. The retrieval is specified by typing the characters '!?', followed by the argument and terminated with a 'carriage return'.

The *single-step-buffer* task involved stepping through the history list an item at a time. To step to the next most recent history item, the 'j' key is pressed. To step to the previous most recent history item, the 'k' key is pressed. These keys are typically alternative forward/backward cursor keys in most UNIX applications. When the desired command appears, selection is made with a 'carriage return'.

The *popup-menu* task involved selecting the generated command from an eight item menu. The menu pops up when the middle mouse button is held down. A menu selection is made by dragging the mouse cursor down to the item, while holding the middle button down, and then releasing the middle button. If the generated command does not appear in the menu, a null selection is made by releasing the mouse button while the cursor is positioned either above or below the menu area.

Procedure

Prior to the start of an experimental session, subjects performed a typing speed test. Each *session* consists of two *practice blocks of test trials* and a *real block of test trials*. The first practice block consists of 4 sets of 9 trials each. In the first set, subjects are presented with each stimulus in the practice block to unjumble and type-in – one at a time from the least recent to the most recent. If an error is made, they repeat this unjumbling and typing task. In the other three sets, subjects practice each of the three history tasks independently. Note, each subject was randomly assigned to a practice order with an equal number of subjects for each practice order.

The second block of trials gives a practice run of the real thing. Subjects are shown each item in the stimulus set, from the least recent to the most recent, in a combined unjumbling and typing task. Then, subjects work through 48 test tasks; 21 of which are typing and the remaining 27 are equally divided amongst the three history-based command specification tasks. The particular cued task is randomly assigned.

[3] Ambiguities relating to this unjumbling step are minimized but some UNIX commands can be ambiguous in the absence of a real task context. Therefore, we tried to choose arguments that would suggest what the appropriate unjumbled command should be (e.g., cat file1 file2 rather than cat file2 file1).

The real block of test trials follows the same procedure as the second practice block but there are 87 test tasks; 39 of which are typing and the remaining 48 are equally divided amongst the three history-based tasks.

Results and Discussion

For each of the four tasks, we generated a task description (see Appendix F). Table 6 lists the predicted performance time associated with each task. Except for S and M_K (determined empirically for each subject), all MHP operators were obtained from the literature (see Table 1). Each subject's keystroke time, M_K, was obtained from a timed typing task (i.e., typing time is divided by the number of characters they typed from a 471 character piece of text). The mean keystroke time was 270 ± 20 (18)[4] msecs. The system response time, S, was measured in the trials for the pop-up menu task. The mean system response time was 220 ± 4 (278) msecs.

In each trial, one of the four tasks is randomly assigned to the subjects. As a result, part of the performance time includes a *choice reaction time*. This is the time required to react to the requested task by initiating the correct task response sequence. Since there are four possible tasks from which the correct response is chosen, reaction time is a function of the time to make one of the four possible choices. According to Card, Moran, and Newell (1983), this choice reaction time is:

$$C_{CRT} = 150 \sum_{t=1}^{4} p_t \log_2 (1 + 1/p_t)$$

where p_t is the probability of an occurrence of task t or equivalently, the fraction of the trials containing task t. Each of the three history-based tasks appears in 16 trials while the typing task appears in 39 trials for a total of 87 trials.

Tasks	Predicted Times
Typing	$P_I + C_{CRT} + (4 + 3a + \sum_{A=0}^{a} n_A)C_E + (1 + a + \sum_{A=0}^{a} n_A)M_K$
Recall by argument	$P_I + C_{CRT} + 2C_M + (n_A + 4)C_E + (n_A + 3)M_K$
Popup menu	$P_I + C_{CRT} + C_M + 2C_E + M_H + M_{D_M}(l) + S + P_S(l)$
Single step buffer	$P_I + C_{CRT} + C_M + P_S(l) + lP_I + (l + 2)C_E + lM_B + M_K$

Table 6: The predicted task performance time as a function of the relevant MHP operators. a is the number of arguments in a UNIX command. n_A is the number of characters in word A of the UNIX command. l is the serial position of the target item. S is the system response time.

[4] The notation 270 ± 20 (18) represents a mean value of 270, a standard error of 20, and a sample size of 18.

A subject's real test trial was discarded if an error was committed in the course of the trial (e.g., incorrect selection, mistyped argument or command name). Each experimental task has a number of different instances related to the parameters of the task. These parameters are obtained from the equation for the predicted times to perform each task (see Table 6).

Typing task: a is the number of arguments in the command line and $\sum_{A=0}^{a} n_A$ is the number of characters in the command line (excluding spaces).

Recall by argument : n_A is the number of characters in the argument.

Popup menu : l is the menu position of the desired command line (note: l = 9 represents an item not on the menu).

Single step buffer : l is the number of steps to the desired command line.

Table 7 lists the various instances of each experimental task in terms of the task parameter values. All subjects' trials were separated into the appropriate task instances. Table 7 lists the total number of trials and error-free trials associated with each task instance. Figure 3 plots the mean observed times for the 27 instances of the 4 tasks as a function of the predicted times. The y = x line provides a reference for comparing the mean observed time to the predicted time.

The measure used to describe the error in the prediction is $e_o(i)$:

$$e_o(i) = \frac{\overline{O}_i - P_i}{\overline{O}_i} \ x \ 100\%$$

where P_i is the predicted time for the *ith* task and $\overline{O}_i$ is the mean observed time for the *ith* task. The magnitude of the difference in observed and predicted times is expressed as a proportion of task performance time (i.e., observed time). This is a more meaningful measure than the absolute difference since different tasks have different task complexity and hence different task performance times.

Our reason for expressing the error in terms of the percentage deviation from the observed time rather than the percentage deviation from the predicted times – Card, Moran, and Newell (1980)'s approach – is that the error measure provides a more accurate assessment of the deviation from the value of interest (i.e., observed time) and hence, the accuracy of the predictions made by a model. John (1988) used the $e_o(i)$ metric in her analysis of the accuracy of the cognitive models generated using the GOMS Model of Immediate Behaviour.

A goal of engineering models is to make predictions averaging within 80% of actual performance. As the data in Table 8 indicate, the prediction errors in 21 instances of the 4 tasks are well within the 20% error criteria. Of the 6 task instances having greater than 20% prediction error, 1 of them is associated with the single-step-buffer task (i.e., l = 1). We conjecture that because of an experimental bias for items at the top of the history list, users have more familiarity with recent command lines and are more adept at handling a history retrieval involving the most recent command line. This skill coupled with a simple response scheme (single stepping using cursor keys) results in a faster task performance time compared to the task performance times of items further down in the history list. This conjecture is supported by a similar result in our menu search

Tasks & Parameters	Values of Parameters for Task Instances	Number of Trials	Number of Error-free Trials
Typing, $(a, \sum_{A=0}^{a} n_A)$	(2, 11), (2, 12), (3,11)	702	673
Recall by argument, n_A	1, 2, 3, 4, 5, 6	288	261
Popup menu, l	1, 2, 3, 4, 5, 6, 7, 8, 9	288	278
Single step buffer, l	1, 2, 3, 4, 5, 6, 7, 8, 9	288	269

Table 7: For each experimental task, the task instances are enumerated in terms of the task parameter values. Also, the total number of trials and the number of error-free trials for each experimental task are presented. Note: In the menu task, an l value of 9 represents an item not appearing on the menu.

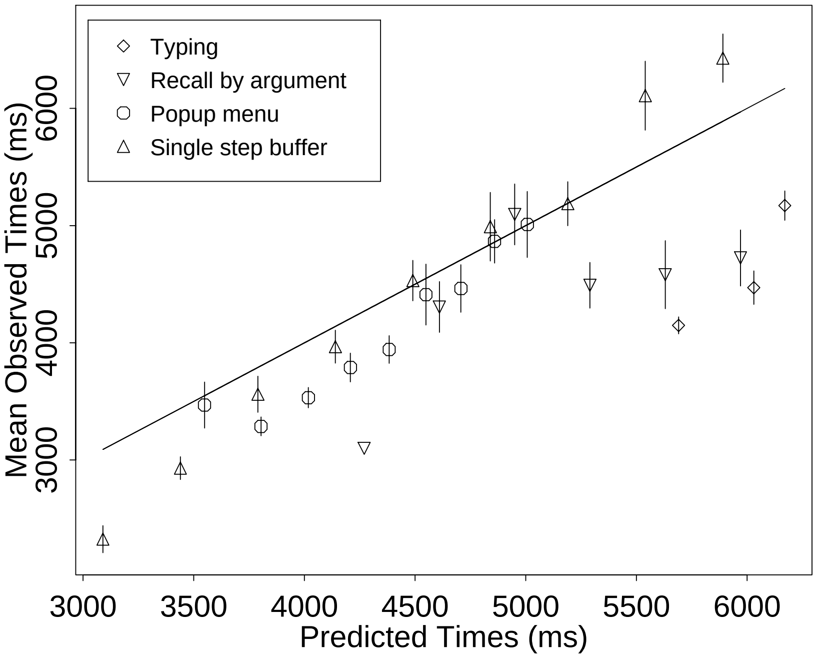

Figure 3: Plot of the mean observed times, along with error bars, as a function of the predicted times. The line is a y = x line.

experiment (see the l = 1 case in Figure 4 of Appendix D). This anomalous behaviour is not evident in the popup-menu task because any user skill with recognizing the most recent item is offset by the less than optimal menu selection behaviour associated with a more complex response scheme. Note, we observed that subjects occasionally made adjustments in the drag operation for over/under-shooting the target. Also, the menu search experiment and the single-step-buffer design use a simple push-button response scheme.

Task	Predicted Times (msecs.)	Mean Observed Times (msecs.) Mean ± SE (N)	% Error $e_o(i)$	>20% Error
Typing				
2 args, 11 chars	5690	4150 ± 70 (361)	-37%	†
2 args, 12 chars	6030	4470 ± 140 (144)	-35%	†
3 args, 11 chars	6170	5170 ± 130 (168)	-19%	
Recall by argument				
1 character	4270	3100 (1)	-38%	†
2 characters	4610	4310 ± 220 (65)	-7%	
3 characters	4950	5100 ± 260 (51)	3%	
4 characters	5290	4490 ± 190 (62)	-18%	
5 characters	5630	4580 ± 290 (41)	-23%	†
6 characters	5970	4720 ± 240 (41)	-26%	†
Popup menu				
position 1	3549	3470 ± 200 (35)	-2%	
position 2	3804	3290 ± 80 (54)	-16%	
position 3	4018	3530 ± 90 (35)	-14%	
position 4	4208	3790 ± 120 (34)	-11%	
position 5	4383	3940 ± 120 (36)	-11%	
position 6	4549	4410 ± 260 (18)	-3%	
position 7	4707	4460 ± 200 (17)	-5%	
position 8	4859	4870 ± 190 (18)	0%	
position 9	5007	5010 ± 280 (31)	0%	
Single step buffer				
position 1	3090	2330 ± 120 (35)	-33%	†
position 2	3440	2930 ± 100 (36)	-17%	
position 3	3790	3560 ± 150 (51)	-6%	
position 4	4140	3970 ± 140 (36)	-4%	
position 5	4490	4530 ± 170 (31)	1%	
position 6	4840	4990 ± 290 (30)	3%	
position 7	5190	5190 ± 190 (15)	0%	
position 8	5540	6110 ± 290 (18)	9%	
position 9	5890	6430 ± 210 (17)	8%	

Table 8: For each of the 27 instances of the four experimental tasks, the predicted times, the mean observed times (along with standard error and number of trials), the error as a percentage of mean observed times and predicted times, and the cases where the error is greater than 20% are presented. M_K is the mean keystroke rate 270 ± 20 (18) msecs.

The other 5 of the 6 task instances having greater than 20% prediction error are associated with the two tasks involving typing (i.e., typing and recall-by-argument). In all cases, the predicted times are longer than the observed times (see Table 8). We conjecture that the higher prediction error is attributed to an inaccurate estimate of the keystroke time M_K. Note, in the interest of simplifying the processing of the subject's data, we decided not to capture the raw keystrokes used in the typing and recall-by-argument tasks. However, the MHP/GOMS framework only deals with error-free behaviour and we need to take into account the possibility that subjects might make typing mistakes and take time to correct them. We accounted for this time by measuring a keystroke time M_K (i.e., time per keystroke) that included a fraction of the time to detect and correct typing mistakes (i.e., we used the mean typing time – the time to type a 471 character piece of text divided by the number of characters in the subject's typed document). In hindsight, this typing task has a number of problems. First, typing a 471 character piece of text is somewhat different from typing a UNIX command line (11 or 12 characters long) or an argument (1 to 6 characters long). Second, there are more instances of typing mistakes, and hence more time spent recovering from typing mistakes, in a typing task involving a 471 character piece of text than in a 11 or 12 character command line. Thus, our estimate of M_K is inflated. It may have been more appropriate to have subjects perform a series of timed typing tasks involving UNIX command lines.

Table 9 lists the mean and range of the absolute prediction error for the 4 tasks by combining all instances of the same task. Both the popup-menu and

Tasks	Number of Tasks	Absolute % Error $e_o(i)$ Mean (Range)
Editing - CMN (404)	12	10% (0% – 37%)
Drawing - CMN (404)	15	19% (1% – 51%)
Computing - CMN (404)	5	24% (9% – 39%)
Combined	*32*	*16% (0% – 51%)*
Command abbreviation - John (36)	3	26% (0% – 71%)
Fitts & Seeger Experiment - John (39)	4	16% (2% – 33%)
Duncan Experiment - John (40)	4	8% (7% – 10%)
Morin & Forrin Experiment - John (42)	6	23% (13% – 38%)
Combined	*17*	*19% (0% – 71%)*
Typing commands	3	30% (19% – 37%)
Recall by argument	6	19% (3% – 38%)
Popup menu	9	7% (0% – 16%)
Single step buffer	9	9% (0% – 33%)
Combined history designs	*24*	*11% (0% - 38%)*
Combined	*27*	*13% (0% - 38%)*

Table 9: The absolute prediction error in 3 different types of tasks (editing, drawing, and computing tasks) examined by Card, Moran, and Newell (1980) (abbreviated as CMN with page numbers in brackets); in 4 different stimulus-response compatibility (SRC for short) tasks examined by John (1988); and 1 command typing and 3 different history tasks that we examined. In the case of Card, Moran, and Newell (1980)'s results, we recomputed the prediction error using $e_o(i)$.

single-step-buffer tasks have a mean absolute prediction error of less than 10% (i.e., 7% and 9%) while the recall-by-argument task has a mean absolute prediction error of 19%. The typing task is the only task that does not meet the 20% error goal.

All three history tasks involve at least one retrieval operation (i.e., retrieval of the task method – see Table 6). A comparison of the absolute prediction errors of the instances of these 3 tasks reveals a range of 0% to 38% and a mean absolute error of 11% (see Table 9). This result indicates that the estimate we used for C_M is an appropriate estimate for the mental retrieval of a task method. Only the recall-by-argument task has an additional mental retrieval operation for selecting the appropriate command argument for the retrieval pattern (see Table 6). Table 8 reveals that the predicted time in all instances of this task is higher than the mean observed time and the range of the absolute prediction error is 3% to 38%. However, because there is a possibility that an inflated estimate of keystroke time M_K may lead us to predict a higher task performance time, we are unable to assess the appropriateness of using the same mental retrieval operator for the second retrieval operation.

An important question is how good the predictions for our command specification task are compared to GOMS predictions for other tasks. We are aware of only two studies which examine the accuracy of the predictions made by the cognitive models generated using the MHP/GOMS framework.

The first study is the experiment to determine how well the Keystroke Level GOMS Model predicts performance times [Card, Moran, & Newell, 1980]. 32 different task-system combinations were examined in the experiment. 14 different tasks (4 editing, 5 drawing, and 5 computing) and 11 different systems (3 editors, 3 drawing packages, and 5 command executives) were considered in the experiment. The mean absolute prediction error for the editing, drawing, and computing tasks are 10%, 19%, and 24%, respectively (see Table 9 for the range of absolute prediction errors). The mean absolute prediction error for the combined group of tasks is 16%.

The second study is a series of experiments to determine how well the GOMS Model of Immediate Behaviour predicts performance times for a number of immediate behaviour tasks [John, 1988]. The group of tasks included a set of 3 command abbreviation recall tasks and 2 different sets of spatial SRC tasks (Fitts & Seeger experiment and Duncan Experiment) and a set of symbolic SRC tasks (Morin & Forrin Experiment) drawn from the literature. John (1988) generated predictions for response time performance for the 4 sets of immediate behaviour tasks and compared them to the observed response time (see Table 9). The mean absolute prediction errors for the 4 sets of tasks are 26%, 16%, 8% and 23% and the mean absolute prediction error for the combined set of tasks is 19%.

Our own experiment examined the accuracy of using both GOMS models (i.e., Keystroke Level Model and the GOMS Model of Immediate Behaviour enhanced to include an estimate for the menu search operator). The mean absolute prediction errors for the 4 tasks are as good if not better than the mean absolute prediction errors of the tasks examined by Card, Moran, and Newell (1980) and John (1988). Compared to the mean absolute prediction errors for the combined group of tasks examined in each of the other two studies (i.e., 16% and 19%), our combined group of tasks has a mean absolute prediction error of 13%.

Remarks about Our Studies

In closing, we make two sets of general remarks concerning the problems of cognitive modelling and the possibilities for future work.

Problems with Cognitive Modelling

The MHP/GOMS framework assumes expert, error-free, routine cognitive-skilled user behaviour and stipulates that predictions are *approximate*. Typically, users are rarely true experts and their behaviours are rarely error-free. This was especially evident in the typing task in our experiment where only one subject was a fast and accurate touch typist. In many instances of the test trials, subjects were observed to be correcting their typing mistakes. Typing errors may be, in part, caused by the fact that subjects were unfamiliar with the keyboard and its layout. Thus, the predicted times represent a lower bound because the observed performance times will typically include non-error-free or suboptimal behaviours.

Recent extensions to the MHP/GOMS framework reveal that not all mental operators are of similar duration as was initially assumed in Card, Moran, and Newell (1983)'s original proposal. Our menu search operator consists of perceptual and cognitive operators. It is an example of a mental operator which is not constant; its value is dependent on where the target item is located in a menu. Also, the designs examined in our study involved a number of different mental retrievals which are assumed to require equivalent duration (C_M = 1200 msecs.). A finer grained analysis and better distinctions between proposed designs would be possible if we could differentiate these various mental operators.

In many places in our task analysis, it was difficult to decide whether certain operators were needed because of a lack of a detailed set of guidelines and rules for doing the task analysis. John (1988) points out that it is often difficult to know what the underlying mechanisms are that are driving the internal processes involved in accomplishing a task and whether the analysis successfully explains and predicts performance. This problem is not unique to MHP/GOMS but permeates all forms of cognitive modelling. In general, current cognitive models are guesses of what the underlying mechanisms are. John (1988) suggests that only by producing more successful cognitive modelling cases can we hope to extract the elements of truth about our guesses and to formulate task analysis rules and guidelines. Of course, user interface designers also need adequate training in cognitive modelling in order to generate not only reasonably accurate predictions but also valid ones.

We also need more comprehensive descriptions about MHP operators because not all operators involve a single act of the MHP processors. In fact, many MHP operators (e.g., $P_S(l)$, C_M, $M_{D_M}(l)$, $M_{P_M}(l)$) are composed of several actions of the MHP processors. Aside from the estimates for these operators, the description should provide an operational definition, as in the case of John (1988) (see pp. 44). Otherwise, user interface designers may inadvertently incorporate the same operator twice in their calculations. A case in point is the operator based on Fitts Law. Experiments to determine the estimates associated with Fitts Law (See Eqn(2)) generally involve a series of reciprocal tapping tasks. Each tapping operation delimits the end of one tap and the start of the next tap. Thus, the elapsed time between two taps inherently incorporates the cognitive and motor operations associated with each tapping task.

Future Work

We took one approach to investigating the issue of user effort. However, because of the emphasis placed by this approach and the simplifying assumptions that were made, there were a number of factors that could not be examined. We discuss possibilities for further analyses.

While task performance times and number and types of MHP operators represent one type of mental-effort measure, mental workload and user errors represent other types. Given the limitations in distinguishing certain mental operations, mental workload may offer a better assessment of mental effort. However, there is very little mental-workload-related work in cognitive modelling (except for [Lerch, Mantei, & Olson, 1989]); the bulk of this research is done in human factors.

The question of which method, history or typing, causes more errors is important as errors generally mean additional effort for error recovery. Because history tools require less physical effort, the potential for making motor errors (typing errors) is reduced, especially for poor typists, but with the increased mental effort, there is more potential for cognitive errors. By the same token, if a memory retrieval in one history-based design requires less effort but tends to be more error-prone (i.e., it retrieves the wrong history item) then this design is not favourable because more overall effort, attributed largely to error recovery, is required. Therefore, it would be informative to examine the issue of user errors but the MHP/GOMS framework does not currently address user errors.

While our analysis did not consider the command generation task, we make some conjectures about how this task influences our results. We conjecture that *typing* would be greatly influenced by command generation. Specifically, users need to expend quite a bit of effort to generate each word in a command and to order the words properly (i.e., according to the Keystroke Level Model, one C_M operator per word in the command). By comparison, we conjecture that users may not need to generate the full command in order to use the proposed history designs. Rather they just need to recognize the command from a menu or generate some part of it so that it can be used as an appropriate criteria (e.g., command name, argument, command number) for retrieving the command. Such recall or recognition operations may take longer than the estimates used in our current study, but certainly not as long as the ones for *typing* . Therefore, if command generation is factored into our user effort analysis, *typing* might perform even worse because of the extra mental retrieval operators. Of course, these conjectures need to be verified by an extended MHP/GOMS analysis.

This study considered the literal reuse of a command which is a natural follow-on to the behavioural study of user recurrences (Chapter 5). It would also be instructive to model and analyze the effort to reuse parts of a previously issued command. However, the main obstacle to such an investigation is the uncertainty surrounding the nature of menu search involving partial matching (i.e., selecting an item that partially matches the search item). As in our literal search case, it is unclear what menu search model is appropriate for partial search or whether the model for partial search is linear. Thus, an experiment must be conducted to determine an estimate for menu search involving partial match.

Concluding Remarks

This chapter examines the mental and physical effort in a command specification task – the task of specifying a recurrent UNIX command. Two different command specification techniques are examined: retyping the entire command or using a history tool. The extended Model Human Processor and GOMS framework is used to model and predict performance for the typing and history methods. The time to perform cognitive and motor operations for each method and the numbers and types of MHP operators involved are used as measures of mental and physical effort. Three different classes of history-based command specification designs are examined. The history-based designs cover existing as well as hypothetical designs. They also cover designs exploiting a user's recall and/or recognition capabilities.

The main observations from this study are:

(1) Compared to *typing* , *command feature* designs favour less physical effort at the expense of more mental effort. They require a range of mental effort (i.e., 1.7 secs to 3.1 secs.); *!name* requires the least mental effort of all history-based designs.

(2) In *history menu* designs, the *static menu* design requires less physical effort than the *popup menu* because it does not require holding a mouse button down for menu selection. Despite the small overhead costs associated with screen real estate for a menu and switch in focus of attention, the benefits of the *static menu* design are reduced performance time, constant feedback of the contents of a history, and no pauses due to the display of a menu.

(3) All *step buffer* designs require pauses for system response; the number of pauses depend on the number of items a user examines. If system response times are negligible, pauses are not a deterrent. These designs minimize either physical effort or mental effort.

(4) Most pure *recall* designs favour less physical effort at the expense of more mental effort. Mental effort is attributed to a memory retrieval of some feature of a desired command (e.g., command number, argument, recency). The design with the least mental effort, *!name* , is the one which retrieves a name from working memory as opposed to long-term memory. Therefore, *recall* designs that require fewer or simpler mental operations as well as fewer physical operations will compete favourably with *typing* , especially when users have average non-secretarial typing skills or better.

(5) Most pure *recognition* designs have selection techniques that use a different input device. Therefore, there is extra overhead (.7 seconds) associated with homing to different input devices and this may be somewhat annoying. However, fast *recognition* designs are superior to *recall* designs for users with poor typing skills (i.e., less than 40 wpm) because they require less physical effort (1 sec.) and less mental effort (compare $690 + 110l$ msecs. for menu search to 1200 msecs. for a mental retrieval operation). Furthermore, *recognition* designs provide ancillary benefits such as visual display of a user history.

(6) The primary operators in the pure *recall* designs are memory retrieval and typing operators. As such, they generally perform better than the pure *recognition* designs when expert and good typists are considered. The bulk of the mental effort in a pure *recognition* design is attributed to visual search; it does not require typing. In relation to the pure *recall* designs, the *recognition* designs using menus are more advantageous for less skilled typists when their constant menu selection times become less significant compared to keystroke times.

(7) Compared to the pure *recall* designs, a *combined recall and recognition* design is neither superior in mental (2.3 secs.) nor physical effort (6.28 M_K). Compared to the two pure *recognition* designs, the combined design requires less mental effort but more physical effort for users with average or worse typing skills (i.e., less than 55 wpm). In general, the combined design in our study provides no substantial benefits compared to the pure *recall* or pure *recognition* designs. This design requires a large variety of operators: cognitive, perceptual, keystroke and system response operators.

(8) *Typing* requires less mental effort (1.2 secs.) than all the proposed history-based designs. It favours less mental effort at the expense of more physical effort; it requires the most keystroke operators (10.5 M_K). Therefore, with less skilled typists, *typing* actually incurs a heavy penalty. In fact, it requires the most effort when compared to all history-based designs when non-typists are involved. An average non-secretarial typist (40 wpm) and a typist unfamiliar with a keyboard can specify a recurrent command faster and with less overall effort using a *static menu* design than *typing* .

In summary, while a history-based design for command specification requires more mental effort, it requires less physical effort when compared to *typing* . The additional mental effort is due to memory retrieval of actions. However, if simpler mental retrieval operations are used (e.g., working-memory retrieval or visual search), history-based designs can compete favourably with typing. The fastest *recall* and *recognition* designs, *!name* and *static menu* , are the ones that minimize mental effort and physical effort.

Our study of the accuracy of the predictions generated by the cognitive model using the MHP/GOMS framework revealed that the predictions are well within the goal of 80% of mean observed performance times. We selected one design from each of the three classes of history-based designs and *typing* examined in the user effort study. The mean absolute prediction error of the three history-based designs is 11% while the typing method had an absolute prediction error of 30%. The higher prediction error for the recall-by-argument and typing tasks are attributed to a poor technique for estimating the keystroke operator M_K. Consequently, the predicted times were much higher than the observed times due to an inflated estimate of the keystroke time. In terms of using the existing estimate for the mental retrieval operator, our empirical data indicates that it is a good enough estimate for the operation associated with retrieving the method for executing the history design. Because our estimate of the keystroke operation may be inflated, we were unable to determine the suitability of using the mental retrieval operator in the recall-by-argument task for choosing a distinctive command argument as a selection pattern. However, the absolute prediction error of this task is 19% and this is within the 20% error goal.