Appendix C

Cognitive Modelling Framework

The cognitive modelling analysis framework used in this thesis consists of the Model Human Processor (MHP for short) and a representation and methodology, known as GOMS, for describing a user's task.

Model Human Processor

Model Human Processor is a general characterization of human information processing (Figure 1); it includes a system architecture and a set of quantitative parameters of component human performance. The system architecture is made up of three interacting subsystems – perceptual, cognitive, and motor – together with a set of ten principles of operation [1] derived from psychological evidence and a human information-processing model of psychological behaviour. Each subsystem has its own memories and processors. The principles of operation describe a wide range of operations and control of these memories and processors.

Each processor operates serially but concurrently with other processors, subject to limitations imposed by data flow requirements. The perceptual processor receives information from the outside world and deposits it into the appropriate sensory memory – visual image store or auditory image store – where it is held until it is symbolically encoded. Working memory receives encoded information which is then used by the cognitive processor, along with previously stored information from long-term memory, to make decisions about what to do. The cognitive processor deposits information into working memory which, in turn, initiates the associated motor processor actions that act on the outside world.

There are quantitative parameters associated with each subsystem component A memory component has three parameters: storage capacity μ, decay time of an item in the memory δ, and main code type (physical, acoustic, visual, or semantic) κ. A processor component has a cycle time τ parameter – time to process one unit of information. In the original model, a parameter value may be expressed in terms of a *typical* value or as a lower and upper bound (i.e., *range*).

[1] Rationality Principle: People attain their goals in a rational manner, given a task structure and inputs, bounded by limitations on their knowledge and processing ability.

Elemental acts of the processors are known as *operators*. A task analysis in this framework involves subdividing a task into the gross functions performed by each processor and decomposing these functions into sequences of operators. Approximate engineering-style calculations of the times to carry out gross-level human behaviours are determined from task descriptions [Card, Moran, & Newell, 1983]. MHP assumes that operators are independent of the task environment so that they can be combined to perform many different tasks. It also 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. Example 1 illustrates a sample task from Card, Moran, and Newell (1983) which shows how a task is modelled and a calculation is made.

GOMS

GOMS is a representation and methodology for describing what a user needs to know in order to perform computer tasks (i.e., a user's cognitive structure). User knowledge is represented in terms of four cognitive components of skilled task performance:

(1) Goals are defined in terms of user tasks to be performed.

(2) Operators are elementary acts of the three processors which result in changes to a user's mental state or to an external state (e.g., pressing a key).

(3) Methods, associated with goals, are procedures for accomplishing a task and are encoded in terms of the operators.

(4) Selection rules are used to choose amongst a number of methods for achieving a particular goal.

A family of engineering-style cognitive models, based on MHP and formulated at different levels of GOMS analysis, has been developed for a number of complex HCI task domains. The Keystroke-Level and Unit-Task Level Models are two such models formulated at the keystroke and unit-task level, respectively [Card, Moran, & Newell, 1983]. Each model predicts the amount of time required by a user to do representative tasks in the modelled task domain.

The Keystroke-Level Model is based on a simplified GOMS analysis which does not require an explicit analysis of goals and selection rules. It is used in situations where it is possible to specify a user's interaction sequence in detail. However, it has two additional restrictions: a task method be given in order for time predictions to be made and predictions apply only to the execution time of a task. This model has been empirically validated and tested for 14 tasks (i.e., 4 editing, 5 graphics, and 5 command executive), 11 different systems (i.e., 3 editors, 3 graphics systems, and 5 command executive subsystems), and 28 users [Card, Moran, & Newell, 1980].

Critical Elements of a Cognitive Model

In order for designers to use such cognitive models, there are a number of critical elements they must possess: calculation, approximation, zero-parameter prediction, learnability and usability by computer designers, and coverage of the total task [Card, Moran, & Newell, 1983; John, 1988]. Calculation supports predictions based on explicit mathematical models of user behaviour with the task as opposed to judgements, designer's experiences, or evaluations. Approximation is the recognition that the processes involved in user behaviour are extremely complex and that real-world calculations for engineering purposes need only include sufficient detail to do the design job. Zero-parameter prediction stipulates that parameters of the engineering models used to make predictions be determined during construction of the model and prior to its application to a new situation. The basic psychology must be built into the model so that the model can be taught to and be usable by computer-system designers who are not trained psychologists or human factors experts. Finally, an engineering model of a specific task domain must provide coverage for the total task by accounting for all first-order effects observed in the task domain.

Shortcomings of Engineering Models

In the initial effort demonstrating how the proposal would be realized, a number of simplifying assumptions were adopted. Some of these assumptions represent shortcomings of current engineering models which are enumerated herein [Card, Moran, & Newell, 1983; Olson & Olson, 1990]:

- The initial model accounts for skilled and error-free user behaviour but it does not consider unskilled user behaviour and the effects of errors, learning and casual usage on user performance.
- The initial model focuses on the usability of a system and does not address the functionality of a system (i.e., what computer tools are needed).
- The initial model uses a mental operator to represent all cognitive processes instead of a differentiated treatment of such cognitive processes which take different amounts of time.
- The initial model assumes that the human information processing components operate in a serial fashion and provides no account of the parallel processes involved in user behaviour.
- The initial model does not address the issue of how much information is held in memory when a computer system is used (i.e., mental workload), of individual differences amongst users, of user fatigue, and of the social and organizational considerations associated with computer-supported work.
- The model does not provide guidance in predicting the usefulness, user satisfaction, and user acceptability of a system.

Recent cognitive modelling efforts are attempting to address a number of these shortcomings (see Olson and Olson (1990) for a review of research in this area) but the work is still very much in its infancy.

Extensions to MHP and GOMS

A number of researchers have extended Card, Moran, and Newell (1983)'s work along two fronts. First, they have confirmed the basic set of parameters of the processors in MHP and extended the number of parameters to include the time to perform component activities for a number of new HCI task domains [Olson & Olson, 1990]. Second, John (1988) has developed two additional GOMS models by extending MHP and GOMS analysis to include two new HCI task domains: stimulus-response compatibility and transcription typing.

Both of these developments have direct implications for the studies in Chapter 6 and Appendix D. MHP's parameters have been extended to include task activities that are particularly relevant to those that underlie the task of using a history tool. In particular, John, Rosenbloom, and Newell (1985) and John (1988) extended MHP to include a class of user behaviour involving operators that are at a lower level than any operators found in previous GOMS models. Furthermore, John (1988)'s work demonstrated how to describe such user behaviours using an algorithmic GOMS formulation. This formulation is adopted as the representation for task descriptions used in this thesis. The following section is a brief presentation of the algorithmic GOMS formulation used in John (1988).

GOMS Model of Immediate Behaviour

Stimulus-response compatibility (SRC for short) is a robust psychological phenomenon where the degree of difficulty of a response is dependent on the complexity of the relationship between the set of stimuli and their corresponding responses [John, 1988]. It was selected as the domain within a class of user behaviour, known as *immediate behaviour*, in which to expand GOMS. Immediate behaviour is the direct mapping of a stimulus into a response without problem-solving or planning [John, 1988]. Such behaviours are examples of routine cognitive skill and are generally associated with reaction time tasks where the response to a stimulus is well known and initiated immediately upon presentation of the stimulus, without deliberation about what response is appropriate.

ARTLESS: An Algorithmic Response-Time Language

Rosenbloom suggested that people perform SRC-related tasks by executing algorithmic programs and proposed an informal, algorithmic response-time language, ARTLESS, for describing the performance of such tasks [John, 1988]. This language was used by John (1988) to describe SRC and transcription typing tasks. A longer description of the language appears in John (1988).

The features of the language are:

(1) Tasks are accomplished with a sequence of MHP processor operations.

(2) Each operation has an identifiable duration that is only dependent on the processor performing the operation.

(3) Tests in conditionals must be exhaustive and explicit because there is no IF-THEN-ELSE construct (i.e., two ARTLESS IF-THEN statements are used instead – testing for the conditional and its negation).

(4) Tests to exit a loop take time only when they succeed.

(5) Reasonable algorithms are purely functional (i.e., there are no null operations or empty loops).

In subsequent descriptions of ARTLESS, an example task description using ARTLESS (Example 2), drawn from our history effort analysis, is used to illustrate the language's constructs. ARTLESS has seven constructs: constants, variables,

Example 2: In the *csh* history tool, a previous command line containing an *argument* is reused by the history command *!?argument*. As a simplification, we will assume that the history tool looks for the most recent occurrence of the command line in the history containing the given *argument* . Furthermore, let us assume that the average length of a typical UNIX command argument is $\bar{n}_A$ characters long. Therefore, an ARTLESS description of the cognitive and motor processes involved in issuing the history command *!?argument* is as follows:

L1	C_M	Get-!? (<!?>)
L2	C_E	Initiate (<!>)
L3	M_K	Key (<!>)
L4	C_E	Initiate (<?>)
L5	M_K	Key (<?>)
L6	C_M	Get-One-Argument ("Argument")
L7		Count ← 1
L8		REPEAT BEGIN
L9	C_E	Initiate (Argument[Count])
L10	M_K	Key (Argument[Count])
L11		Count ← Count + 1
L12		UNTIL Is-Equal? (Count, $\bar{n}_A$)
L13	C_E	IF-SUCCEEDED Is-Equal? (Count, $\bar{n}_A$) THEN BEGIN
L14	C_E	Initiate ()
L15	M_K	Key ()
L16		END

assignments, blocks, operators, branches, and loops. Constants, variables and assignments are like typical programming language constructs (e.g., 1, Count, Count ← Count + 1). Blocks – beginning with the keyword BEGIN and ending with the keyword END – delineate sequences of operations (see lines L13 - L16).

Operators are parameterized black boxes that do the real work in the system (e.g., Get-!? (<!?>) retrieves the method for the !?argument task). Lines L1-L6, L9-L10, and L12-L15 contain operators. Predicates are a subclass of operators that don't return a value but are used to test for the truth of some condition (e.g., Is-Equal? (Count, $\bar{n}_A$) test for the truth of equality of the two arguments – Count and $\bar{n}_A$). Lines L12 and L13 contain predicates. Algorithms are specified without the need to detail how the concept "Argument" is represented.

Branches permit the testing of a predicate and the subsequent execution of an operator or a block of operators when the predicate is true. Lines L14 and L15 are executed only if the variable Count and the constant $\bar{n}_A$ are equal (see L13). Since, an ELSE clause is not explicitly supported, a branch requiring such a clause may be obtained by an explicit test for the complement of the IF clause.

Finally, a loop executes an operator, or block of operators, until some predicate becomes true (see Lines L8 - L12). There must be an explicit test for the predicate at the end of the loop – line L12 is the explicit test for the REPEAT loop in line L8 while the predicate test in L13 is provided for readability.

Cost Scheme for an ARTLESS Algorithm

A simple scheme is used to assess the cost of an algorithm.

(1) Except for operators, all constructs in ARTLESS are free of charge. In Example 2, lines L7, L8, L11, L12, and L16 are free because they do not involve the operator construct.

(2) Each operator takes one unit of time associated with the processor that performs the operation. The operator times are expected to relate to the respective processor cycle times for the MHP. Therefore, lines L2, L4, L9, L13, and L14 involve the cognitive processor (note the C_E adjacent to the lines), while lines L3, L5, L10, and L15 involve the motor processor (note the M_K adjacent to the lines). Both lines L1 and L6 involve operations that are not elemental operations of the cognitive processor and are each assigned a composite cost of C_M. Line L1 requires a retrieval of the history command (i.e., *!?*) from Long Term Memory while line L6 selects one of the arguments in the desired command line as the pattern to the history command.

(3) The REPEAT loop is a special case for counting operators. All operators within the loop count as they are executed. The predicate in the UNTIL line of the loop does not count, but the predicate outside the loop – which retests the termination of the loop – does count. This is why line L12 has no cost assigned to it but the predicate test in L13 does, to represent a one time cost for the REPEAT loop in lines L8 - L12.

Appendix D
Visual Search of a History Menu

This appendix presents an experiment to estimate a MHP operator required for our analysis of the user effort required to issue a recurrent user operation (see Chapter 6). The reason for the estimation experiment is that the MHP/GOMS framework does not model visual search of a menu and is therefore unable to predict user behaviour involving this task. Furthermore, this information is not available in a form which can be incorporated into the framework. The reason is that there are conflicting theories about the visual search strategy that people use.

Estimation of Visual Search Operator for a History Menu

Studies in cognitive psychology have shown that when a sequence of alphanumeric characters is displayed in the foveal (eye focus) region of the eye (eliminating the need for eye movement), an exhaustive search strategy is used (i.e., each element is examined). On the other hand, when a large visual list is involved, necessitating eye movements, a self-terminating search strategy is observed (i.e., search terminates ones the search item is found) [Chase, 1986]. Typically, a menu is sufficiently large to necessitate eye movements so that a self-terminating search strategy would be in effect.

On the other hand, studies of menu search within HCI using real-world stimuli have suggested that knowledge (e.g., menu organization and experience) plays a role in directing and limiting search of menus [Lee & Raymond, to appear; Mehlenbacher, Duffy, & Palmer, 1989]. Specifically, users can narrow the search space to a smaller area using knowledge about the menu. As a result, menu parameters such as organization, frequency of reference, match criteria (e.g., literal, categorical, partial) could play a role in narrowing the search space and influence the visual search strategy used [Giroux & Belleau, 1986; Paap & Roske-Hofstrand, 1988].

Current menu search studies have suggested two candidate models: a systematic self-terminating search and a random search [Card, 1984; MacGregor & Lee, 1987a]. In a systematic self-terminating search, users use some strategy (e.g., top down) to keep from examining an item that has been previously examined and users terminate the search process as soon as the examined item matches the search item (i.e., target item). This search strategy is based on empirical studies of search times for categorical information (i.e., search that involves matching a target with its appropriate category). In contrast, items are examined randomly in a random search, some having been examined previously, until the target item is located. This search strategy is based on empirical studies of search times and eye movement data for visual search of command menus organized in one of three ways: function, alphabetic, and random. There is currently some controversy as to whether the current data differentiate between the two models [Card, 1984; MacGregor & Lee, 1987b]. Aside from differences in strategies, the two models would also predict different search times.

Given the lack of a unified visual search theory as well as a lack of a clearcut visual search strategy for our situation, we ran an experiment to determine the strategy for our history-menu search task. There were two objectives for this experiment. The first objective was to determine the strategy used in a visual search of a history menu. The second objective was to estimate the time to perform this task for the menu search operator.

Method

Our experiment is modelled closely after Neisser (1964)'s visual search paradigm. It is two independent experiment conducted together. Each experiment is a two-factor within-subjects design measuring visual search performance with a history menu. However, menu references exhibited *locality* in the first experiment and *non-locality* in the second experiment. Search items were always present on the menu in the locality experiment and were either present or absent on the menu in the nonlocality experiment.

Serial position is a factor which designates where a search item appears in the menu. Serial position 1 is the top item in the menu and serial position 8 is the bottom item in the menu. *Scenario* is a factor which emulates either the locality or non-locality menu reference behaviour using different constraints and probability distributions. The probability distributions represent the likelihood of finding a search item at each menu position. Two different scenarios are considered in each experiment.

Subjects

16 paid volunteers, consisting of students and staff at the University of Toronto, participated in the experiment. All subjects had knowledge and experience with UNIX. Subjects served in both experiments in one session. This is a deliberate attempt to mirror a typical user session in which their interactions exhibit both locality and non-locality behaviours (see Chapter 5).

In each half of the session, subjects performed one of the scenario conditions from each of the two experiments. No two scenario conditions from the same experiment were administered in the same half of the session. Given this restriction, there were 16 possible combinations for the administration of the scenario conditions. The administration of the scenario conditions were counterbalanced to eliminate possible biases resulting from sequencing effects by randomly assigning a subject to one of the 16 possible combinations.

Materials and Apparatus

A list of 12 items was used with the 8 most recently referenced items being visible on the menu (i.e., the top eight items in the list are visible while the bottom four items in the list are never displayed). The term *menu* is used to refer to the visible portions of the list. The most recently referenced item appears at the top of the menu and the 8th most recently referenced item appears at the bottom of the menu.

The stimuli are UNIX command lines. Table 1 lists the menu items used for each scenario factor in the two experiments. All the items are of the same length because our primary objective is to study how locality and non-locality effects influence visual search in the absence of other influencing factors.

The experiment machine is a SUN 3/50 workstation consisting of a high-resolution graphic display and keyboard. A simple window manager MGR from Bellcore is used. The SUN display is divided into three windows (see right screen panel of Figure 1). The top window contains instructions to the subjects. The bottom window displays an error message whenever the subject makes a mistake (e.g., selecting an inappropriate key on the keyboard). The middle window is the primary interaction window where a list item, the menu, or a search item is presented and subject responses are made.

Scenarios in Locality and Non-Locality Experiment

In the locality experiment, two scenarios were examined: *small locality set* and *large locality set*. Search items in the small locality set scenario appear primarily in the top 4 menu positions, leading to locality set sizes of 1 to 4. On the other hand, search items in the *large locality set* scenario appear in all 8 menu

Locality Experiment		Non-Locality Experiment	
Small Locality Set	Large Locality Set	Mild Non-Locality	Strong Non-Locality
`psroff -ms all`	`dbx search core`	`compress demos*`	`talk alee@db`
`refer paper.ms`	`man -k debugger`	`uncompress pics`	`rlogin front`
`spell abstract`	`search ins outs`	`rcp db:mbox tmp`	`ftp explorer`
`emacs paper.ms`	`ls -lt ins outs`	`grep talks mbox`	`mail -f mbox`
`wc conclusions`	`cd ~/tests/data`	`vi mbox oldmbox`	`ping spadina`
`lookbib biblio`	`mv a.out search`	`tar -f mail new`	`alias p more`
`sortbib biblio`	`cc  -g search.c`	`chmod o-rwx tmp`	`which whoami`
`whereis psroff`	`lint ~/search.c`	`rm mail/db/jan*`	`rn -n can.ai`
`webster albeit`	`jove ~/search.c`	`file mail/db/a*`	`head dumplog`
`cp intro saved`	`cat -n search.c`	`zcat pics pics0`	`stty new crt`
`indxbib biblio`	`lpr -Pntx outs0`	`du -s mail/db/*`	`mkdir backup`
`p intro biblio`	`diff outs outs0`	`tail -f oldmbox`	`rlog letters`

Table 1: The menu items used in the locality and non-locality experiment.

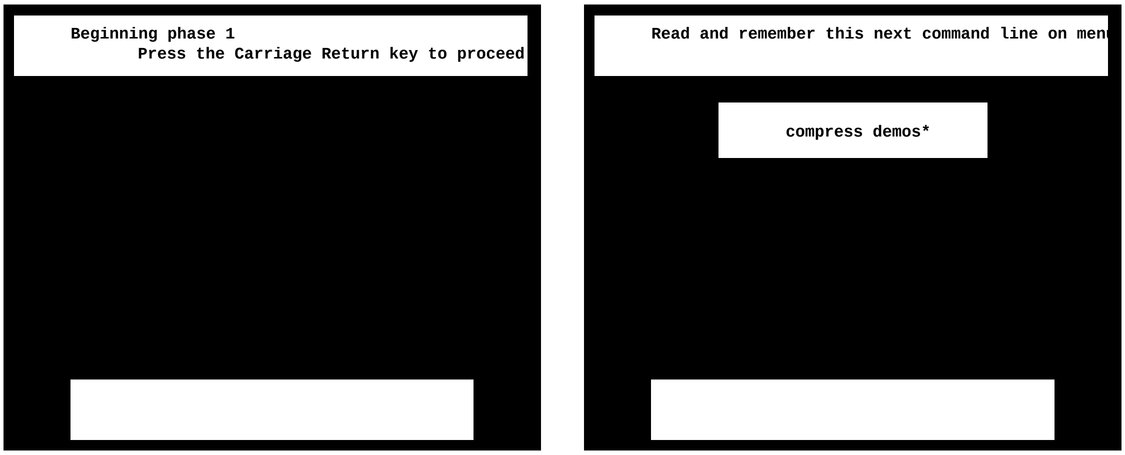

Figure 1: The two screen panels for the priming stage. The left screen panel marks the beginning of the priming stage. The right screen panel is representative of the screen panel when the list items are individually displayed for 3.5 seconds for the user to remember.

positions with diminishing probability, leading to locality set sizes 1 to 8. Table 2a lists the probability of finding a search item in each of the 8 serial positions for the two scenarios in this experiment.

In the non-locality experiment, two scenarios were examined: *mild non-locality* and *strong non-locality*. In the mild non-locality case, search items are absent 25% of the time and in the strong non-locality case, search items are absent 50% of the time. Serial position 0 represents an absent search item. The least recently referenced item is used as the search item in a trial requiring an absent search item. When the search item is present on the menu, the probability of finding it at serial position 2 to 8 is .28, .20, .16, .12, .10, .08, .06. However, since the percentage of the trials where the search item is present is different in the two scenarios, the effective probability of observing a search item is .75 and .50, respectively. Table 2b lists the effective probability of observing a search item in each of the serial positions of the menu. To ensure that no locality set biases occur in this experiment (i.e., no repeated reference to a set of the menu items), the probability of referencing a search item in position 1 is 0. As a result, there are 8 serial positions: positions 2 through 8 in the menu when the search item is present and position 0 when the search item is absent.

The probability distributions used in the scenario conditions were chosen to produce the desired locality and non-locality effects, subject to the constraint that there were sufficient test trials for each serial position.

Procedure

A *session* consists of a *practice block of test trials* and *4 blocks of test trials*. Each block of trials begins with a *priming stage* (see left screen panel in Figure 1). Each of the 12 list items is presented individually for at least 3.5 seconds in the

Locality	Probability of Target at Serial Position							
	1	2	3	4	5	6	7	8
Small Locality Sets	.26	.24	.22	.20	.02	.02	.02	.02
Large Locality Sets	.18	.17	.15	.14	.12	.10	.08	.06

Table 2a: The probability of locating the target in serial positions 1 through 8 for the two scenarios in the locality experiment.

Non-Locality	Probability of Target at Serial Position							
	0	2	3	4	5	6	7	8
Mild Non-Locality	.25	.21	.15	.12	.09	.075	.06	.045
Strong Non-Locality	.50	.14	.10	.08	.06	.05	.04	.03

Table 2b: The probability of locating a search item in serial positions 0 and 2 to 8 for the two scenarios in the non-locality experiment. Serial position 0 represents the situation in which a search item does not appear on the menu.

reverse order of their initial position in the list, least recent first and most recent last (see right screen panel in Figure 1).

The priming stage is followed by a *searching stage* which consists of 100 timed trials (see top left and bottom right screen in Figure 2). On each trial, the search item is presented and is followed 2.5 seconds later by the presentation of the menu (see the top right and bottom left screen panels in Figure 2). If the search item appears in the menu, subjects are instructed to respond by hitting either the ‘f’ or ‘j’ key on the keyboard; whichever delimits the search item in the menu. This simple response scheme is used because it is crucial that the time to locate the item be accurately measured. In each trial, ‘f’ or ‘j’ is randomly assigned to each item of the menu, with equal probability. If the item is not located, subjects are instructed to respond by hitting the space bar on the keyboard. The time between the presentation of the menu and the subject’s response is recorded and any errors that occur in the trial are noted.

Results and Discussion

The following sections describe the results and analyses of each of the two measured scores: number of errors and search times. These results are analyzed, for each experiment, using a two-factor, within-subjects ANOVA test.

Number of Errors

On average, less than 2.5 errors were committed in the 100 timed trials for each scenario condition, collapsed over all serial positions. The mean and standard error of this measure for each of the scenario conditions appear in Table 3.

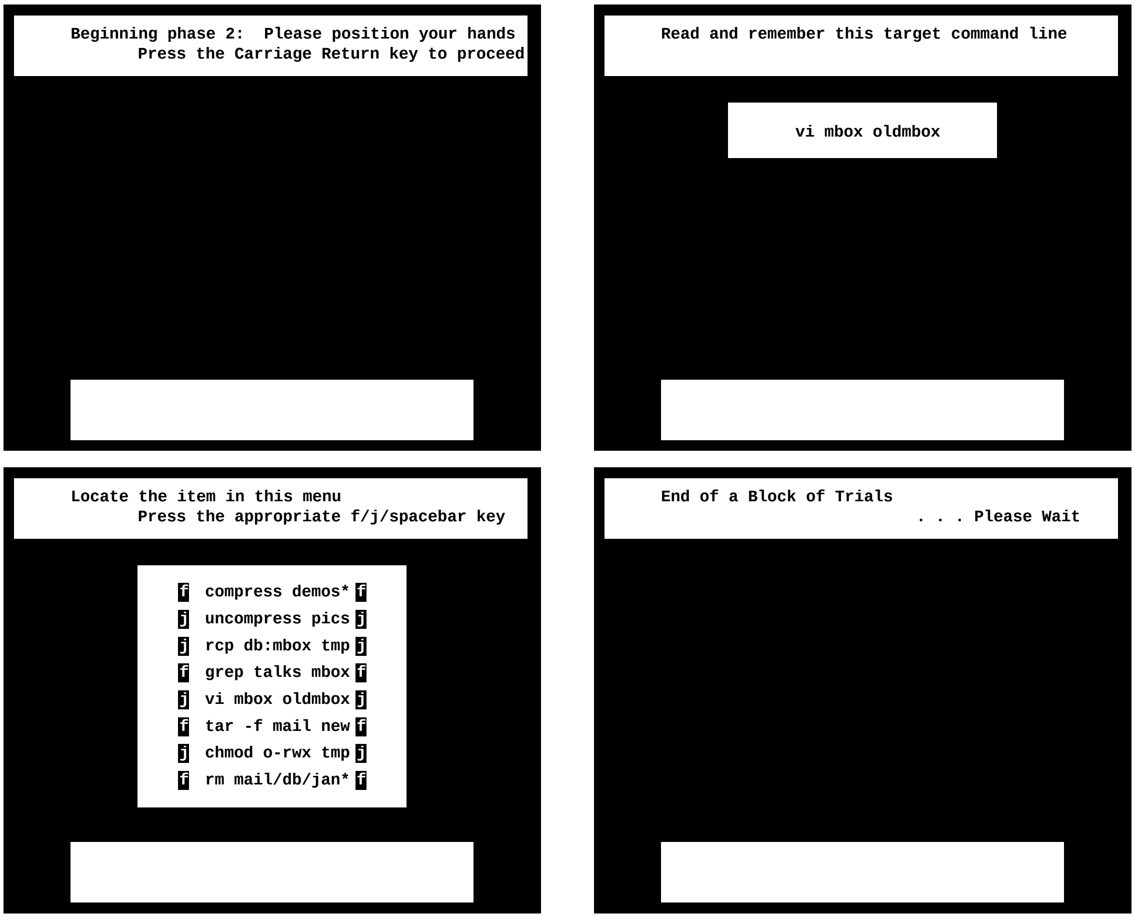

Figure 2: The four panels associated with the searching stage. The top left panel and the bottom right panel mark the beginning and end of the searching stage. The top right panel is representative of the screen panel in which the search item to be searched for is presented for 2.5 seconds. The bottom left panel is representative of the screen panel in which the subject is to locate the search item in the displayed menu. If the subject locates the search item in the menu, the key for the letter delimiting the search menu item is pressed, otherwise, the space bar key is pressed.

There is no significant difference in the number of errors between the two levels of the scenario factor in either experiment (i.e., $F(1,15) = 1.98$ $p = 0.18$ for the locality experiment and $F(1,15) = 0.65$ $p = 0.43$ for the non-locality experiment).

In the locality experiment, the number of errors is significantly different for different serial positions, $F(7,105) = 4.00$, $p < .001$. This result is erroneous and can be attributed to unequal sample sizes in the different conditions (i.e., fewer trials in serial positions 5 through 8 for the small locality set scenario compared to the large locality set scenario). A better estimate of search times in serial positions 5 through 8 is obtained for the small locality set scenario compared to the large locality set scenario.

Locality Mean ± Std. Err. (N)		Non-Locality Mean ± Std. Err. (N)	
Small Locality Sets	Large Locality Sets	Mild Non-Locality	Strong Non-Locality
2.4 ± 0.4 (16)	1.7 ± 0.4 (16)	2.4 ± 0.4 (16)	1.9 ± 0.5 (16)

Table 3: The mean and standard error of the number of errors observed for the 16 subjects in each of the scenario factors in the two experiments.

Search Times

For cognitive modelling purposes, it is sufficient to examine the average visual search performance profile (i.e., to examine average subject performance times). In the same spirit as other cognitive modelling research based on MHP, an approximation of the search strategy and the quantitative relationship between search times and serial position are determined. Therefore, the focus is on the mean search times. This is based on the mean search times of each subject at each serial position in each of the scenario conditions.

Figures 3 and 4 plot, for the locality and non-locality experiment, the mean search times as a function of the 8 serial positions for each of the two scenario conditions in each experiment. As suggested by the respective plots, the main effect of serial position on search time is highly significant in both the locality ($F(7, 105) = 60.6$, $p < .0001$) and the non-locality experiments ($F(7, 105) = 47.9$, $p < .0001$). This suggests that a relationship exists between search time and serial position of search items. In order to determine the exact nature of this relationship, a one-factor within-subjects trend analysis was performed on each scenario condition in each experiment.

In the locality experiment case, there is a significant linear relationship in the small locality set scenario ($F(1, 105) = 134.9$, $p < .0001$) and in the large locality set scenario ($F(1, 105) = 448.2$, $p < .0001$). However, there is also a significant quadratic relationship in the large locality set scenario ($F(1, 105) = 14.0$, $p < .01$). A closer examination of the plot suggests that this may be caused by the low search time for serial position 1. This low search time for serial position 1 suggests that a subject's skill in recognizing the same search items which appear in back-to-back trials is more advanced than their skill in recognizing the same search items which are separated by more than one trial. Thus, subjects may just proceed to make an immediate response without the need to examine the menu items. To confirm this conjecture, a trend analysis was performed again on the data but ignoring the search times for serial position 1. The test reveals a significant linear relationship for each scenario condition (both in the small locality set scenario, $F(1, 90) = 66.7$, $p < .0001$ and in the large locality set scenario, $F(1, 90) = 216.8$, $p < .0001$) and the lack of a significant quadratic relationship in either scenario. We therefore, treat serial position 1 in both scenarios as special cases. This special treatment of serial position 1 is applied to both scenarios in order to ensure equal treatment.

A trend analysis was also performed on each scenario of the non-locality experiment for search times in serial positions 2 through 8. Serial position 0 is

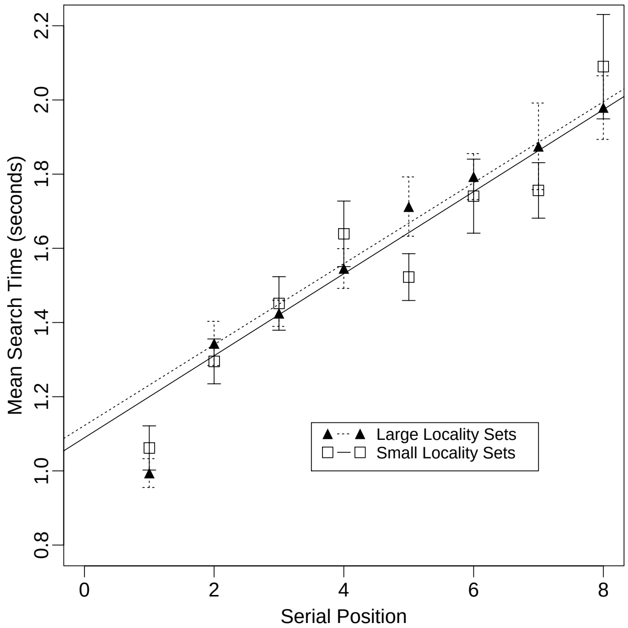

Figure 3: Locality: A plot of the mean search times (based on the mean search times of the individual subjects) as a function of serial position along with the fitted line. Note: the data for serial position 1 was not included in the data used to derive the fitted line because a strong practice effect in this serial position produced a depressed reaction time.

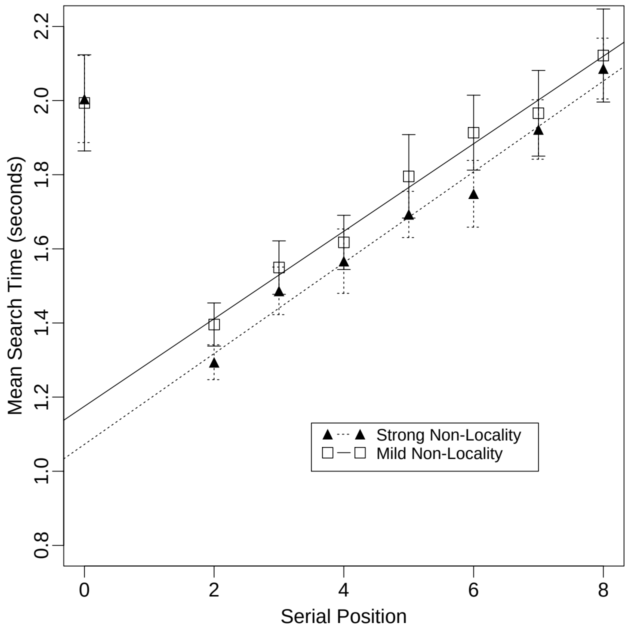

Figure 4: Non-Locality: A plot of the mean search times (based on the mean search times of the individual subjects) as a function of serial position along with the fitted line.

treated as a special case and is not included in the trend analysis because a subject's response in this case is different from the case when the search item appears on the menu. In fact, the response scheme in the latter case is simpler and requires less time than the former case. This is confirmed in an examination of search times for this serial position when compared to search times at the other serial positions. For both scenarios, there is only a significant linear relationship between search times and serial position (in the mild non-locality scenario, $F(1, 90) = 200.1$, $p < .0001$ and in the strong non-locality scenario, $F(1, 90) = 187.3$, $p < .0001$).

Table 4 lists the mean search times for the special cases in each of the two experiments. Based on the results of the trend analysis, we performed a least squares linear regression on the search times against serial positions 2 through 8 for each scenario condition in the two experiment (see Figures 3 and 4). From the regression, the slope and the intercept for each fitted line as well as the coefficient of determination [1], r^2, is determined (see Table 5).

There are four fitted lines, one for each scenario condition.

$$\text{Task Performance Time} = \text{Search Rate} * l + \text{Aggregate Time} \qquad \text{Eqn. (1)}$$

The slope of the fitted line is the incremental time needed to search each additional menu item (i.e., Search Rate). The intercept is the Aggregate Times of the processes that occur in every test trial (e.g., start and stop times of visual scan, decision time, response time). As shown in Table 5, search rates remain relatively unchanged across the two scenario conditions in each experiment. Furthermore, search rates between the two experiments are nearly identical (.11 secs./item in the locality experiment and .12 secs./item in the non-locality experiment). The main differences are attributed to intercept values. This means that each scenario condition involves cognitive operations of different durations (see Table 5).

The results of this experiment indicate that the time to search a history menu is approximated by a linear function that varies directly with the number of items to search (i.e., the menu search is serial). The search rate for both the locality and non-locality experiments are nearly identical with aggregate times that are slightly different for the different scenario conditions. The history lists in our experiment are typical history lists. Therefore, the results are generalizable to user histories whose references exhibit locality and non-locality effects.

Time Parameter for the Pure Menu Search Operator

Eqn. (1) includes a menu search component and a response component. This response component appears in the intercept – Aggregate Time – because the response task is one of the subtasks that is performed in every test trial. Because the analyses in Chapter 6 consider task proposals which use different response methods, the response time in the menu search task is separated out from the task performance time to yield a pure menu search time.

[1] r^2 represents the proportion of the sum of squares of deviations of the y-values about their mean that can be attributed to a linear relationship between y and x.

Experiment	Scenario	Serial Position	Search Times (secs.) Mean ± Std. Err.
Locality	Small Locality Set	1	1.06 ± .06
	Large Locality Set	1	.99 ± .04
Non-Locality	Mild Non-Locality	0	1.99 ± .13
	Strong Non-Locality	0	2.00 ± .12

Table 4: The mean search times, along with the standard error, for the special cases of the locality and non-locality experiment.

Experiment	Scenario	Search Rates (secs/position) Slope ± Std. Err.	Aggregate Times (seconds) Intercept ± Std. Err.	r^2
Locality	Small Locality Set	.11 ± .02	1.09 ± .10	.87
	Large Locality Set	.11 ± .01	1.12 ± .03	.99
Non-Locality	Mild Non-Locality	.12 ± .01	1.18 ± .03	.99
	Strong Non-Locality	.12 ± .01	1.07 ± .04	.98

Table 5: The slope and intercept of a weighted regression fit on the data shown in Figures 3 and 4 for each scenario factor. The slope is the search rate and the intercept is the aggregate duration of the processes that occur regardless of serial position.

Figure 5 contains a ARTLESS description of the menu search task. This description yields another form of Eqn. (1) for the time to perform this task:

$$\text{Task Performance Time} = P_{S_0} + lP_{S_1} + p_f(P_I + 3.5\,C_E + M_B) + (1 - p_f)(3C_E + M_B)$$

where p_f is the probability of the search item being on the menu while $(1 - p_f)$ is the probability of the search item not being on the menu. In the two scenario conditions of the locality experiment p_f is 1. In the non-locality experiment, p_f is .75 for the mild non-locality condition and p_f is .5 for the strong non-locality condition. These p_f values are not in one-to-one correspondence to those observed in Chapter 5. The reason is that, unlike a real user session, a laboratory experiment is a small scale simulation of a user session. As a result, we had to amplify the values for p_f in order to reproduce, in a laboratory experiment, the locality and non-locality effects observed in our behavioural studies (i.e., small and large locality set effects and mild and strong non-locality effects).

$P_S(l)$ is the time associated with the non-response activities. P_{S_0} is the Aggregate Time – Response Time and P_{S_1} is the Search Rate.

$$P_S(l) = P_{S_0} + lP_{S_1}$$

L1	$P_S(l)$	Match_Result ← Search (Target)
L2	C_E	IF-SUCCEEDED Found? (Match_Result) THEN BEGIN
L3	P_I	Letter ← Perceive-Letter (<LETTER>)
L4	C_E	IF-SUCCEEDED Is_F? (Letter) THEN BEGIN
L5	C_E	Initiate (<LETTER>)
L6	M_B	Key (<LETTER>)
L7		END
L8	C_E	ELSE IF-SUCCEEDED Is_J? (Letter) THEN BEGIN
L9	C_E	Initiate (<LETTER>)
L10	M_B	Key (<LETTER>)
L12		END
L13		END
L14	C_E	ELSE IF-SUCCEEDED Not_Found? (Item) THEN BEGIN
L15	C_E	Initiate (<SPACE>)
L16	M_B	Key (<SPACE>)
L17		END

$$
\begin{aligned}
\text{Total Cost} &= P_S(l) + p_f((C_E + P_I + .5(2C_E + M_B) + .5(3C_E + M_B)) \\
&\quad + (1 - p_f)(3C_E + M_B) \\
&= P_S(l) + p_f(P_I + 3.5C_E + M_B) + (1 - p_f)(3C_E + M_B) \\
&= P_{S_0} + lP_{S_1} + p_f(P_I + 3.5C_E + M_B) + (1 - p_f)(3C_E + M_B)
\end{aligned}
$$

Figure 5: A description of the menu search task and the associated performance time. The response component of the task is described by lines L2 to L17. The 'f' and 'j' letters appear with equal probabilities (i.e., .5). A response involving either the 'f' or 'j' key takes one of two routes: a) positive test for presence of 'f' character first (L4 to L7) or b) negative test for presence of 'f' and then a positive test for presence of 'j' (L4, L8 to L12). p_f is the probability of the search item being on the menu while $(1 - p_f)$ is the probability of the search item not being on the menu. There are two routes for determining the correct response: a) positive test for presence of menu item (L2 to L13) or b) negative test for presence of menu item and then positive test for absence of menu item (L2, L14 to L17). p_f is 1 for both scenario conditions of the locality experiment. In the non-locality experiment, p_f is .75 for the mild non-locality condition and .5 for the strong non-locality condition.

The response time is made up of two component response times; one for each of the two selection schemes. The first selection scheme is used when the search item is on the menu and the subject must press either the 'f' or 'j' key, whichever appears next to the search item. The performance time is $P_I + 3.5C_E + M_B$. The second selection scheme is used when a search item is not on the menu and the subject must press the 'space bar'. This performance time is $3C_E + M_B$.

All subjects are familiar with the keyboard so M_B is equivalent to the motor processor cycle time – 70 msecs. Similarly, one perceptual processor cycle, P_I = 100 msecs., is needed to notice the letter delimiting the target. Finally, each mental execution step involves one basic cognitive cycle C_E = 70 msecs.

The resulting response time is 420 msecs when the search item appears on the menu and 280 msecs when the search item does not appear on the menu. Table 6 lists the aggregate time, the response time, and the pure menu search time for each scenario condition in each of the experiments.

Concluding Remarks

Our experiment to estimate the menu search operator considered visual search performance with a menu organized in a least-recently used (LRU for short) order; the top-most menu item is the most recently referenced command while the bottom-most menu item is less recently referenced. Menu references in the experiment exhibited either locality or non-locality behaviours. Results reveal that menu search times are approximated by a linear function of the serial position of the search item in the menu.

Because we did not run subjects for all combinations of the two scenario conditions in the locality case with the two scenario conditions in the non-locality case, we cannot make a statistical comparison between the results of the locality and non-locality cases. However, an informal comparison of the menu search rates for the locality and non-locality case reveals nearly identical values. Any differences in menu search times within the two scenario conditions of the locality and non-locality cases are attributed to nuances in the particular scenarios resulting in cognitive operators of different durations [Chase, 1986].

Experiment	Scenario	Aggregate Time (msecs.)	Response Time (msecs.)	Pure Search Time (msecs.)
Locality	Small Locality Set	1090	420	$670 \pm 110\ l$
	Large Locality Set	1120	420	$700 \pm 110\ l$
	Mean	1110	420	$690 \pm 110\ l$
Non-Locality	Mild Non-Locality	1180	380	$800 \pm 120\ l$
	Strong Non-Locality	1070	320	$750 \pm 120\ l$
	Mean	1130	350	$780 \pm 120\ l$

Table 6: The Aggregate Time from Table 5, the response time, and the pure search time associated with each scenario condition of each experiment and the average case.