Chapter 3

Cognitive and Behavioural Issues

According to Carroll, Kellogg, and Rosson (1991), current design practices in human-computer interaction involve a ''cycle of discovering, defining, and understanding tasks people want to do, need to do, or might do, and then designing, developing, and analyzing artifacts to support tasks'' (pg. 98). Such development activities are circumscribed by a task-artifact cycle which refines, emulates, and synthesizes HCI artifacts based on characterizations, analyses, and comprehension of tasks and artifacts during use. They suggest that the task-artifact cycle is a manifest structure of HCI technology evolution in which tasks and artifacts co-evolve: ''a task implicitly sets requirements for the development of artifacts to support it; an artifact suggests possibilities and introduces constraints that often radically redefine the task for which the artifact was originally developed'' (pg. 79).

The preceding and current analyses are part of an iteration through such a task-artifact cycle. Based on an examination of history-tool artifacts, the preceding chapter provides a rational reconstruction of seven independent tasks which early designers envisioned history tools supporting. It reveals that current history tools focus on providing support for individual uses of history information while paying little or no attention to the integration of various uses of history tools.

This chapter is concerned with the integration of the seven uses of history information to evolve new history-tool designs. The analysis focuses on cognitive and behavioural considerations for such integrated history tools. The first section proposes different kinds of information that a history should maintain. The second section proposes system capabilities needed for using this information.

Contents of a User History

The discussion, thus far, has been rather vague about the kinds of information maintained in a user history except to say that it is a collection of information associated with a user's past interactions with a system. This was done deliberately to allow an unrestricted interpretation of a history use to dictate the kinds of information that a user history should maintain rather than to allow a haphazard choice of whatever is easily accessible.

The seven history uses provide insights into a variety of information that history tools should maintain. While certain history information may serve a number of different purposes, it is unlikely that the particular information can service all seven history uses. Similarly, information available in one form may not be as useful as the same information represented in a different form. Therefore, it would be useful to maintain a variety of information related to a user's past interactions as well as a number of different representations of the same information. In the following subsections, we present a number of distinctions regarding different representations and different kinds of history information:

- Actions and objects referenced in actions
- User inputs and system responses
- States as well as transitions
- Specific information and peripheral information
- Externalized information and internalized information
- User and behavioural information

Actions and Objects Referenced in Actions

In addition to reusing previous actions, users may reuse the objects referenced in their previous actions (e.g., the name of a directory). Many history tools in UNIX maintain a log of user actions only (i.e., whole command lines) and do not keep separate logs of action objects (e.g., file names, directories, network addresses). Such information would facilitate search and selection. For example, the SYMBOLICS GENERA programming environment [McMahon, 1987] supports a type mechanism that maintains multiple lists of different objects using a *user interface type* and allows users to access easily a relevant list, where relevancy is determined by the desired type in the current context.

User Inputs and System Responses

History for inter-referential I/O also reveals that users, in addition to referencing previous user inputs, may also reference system responses produced from previous user actions. Maintaining a history of system responses is useful because users can consult or make use of the system responses in their subsequent

98 OIS.DB% ls /usr

bin/	dict/	hosts/	local@	new/	sccs/	stand/
crash/	etc/	include/	man/	pub/	skel/	tmp/
Diag@	games@	lib/	msgs/	preserve@	spool/	ucb/

99 OIS.DB% !!/skel/

Example 1: The 98th command reveals the contents of a directory, */usr*, which represents information generated as a result of a user request. One piece of the information, *skel/*, is used as part of a subsequent user command which requests the system to list the contents of the subdirectory *skel/*.

actions (see Example 1) without first having to repeat the user actions to derive the desired system response (i.e., history tools can economize user actions).

States As Well As Transitions

A distinction arising from our taxonomy is that a history can be a collection of visual snapshots of system states or a collection of state transitions (i.e., user operations used to effect state transitions). The visual snapshots contain information about particular system states. Such information is useful for helping users to re-acquire the mental context for a desired system state. However, a state-based user history does not maintain explicit information about the operations that led to a state from a previous state. Thus, it is not a suitable representation when users need access to the transition information.

Of course, there are also arguments to be made about the unsuitability of transition-based user histories. To illustrate, users need to examine some aspect of an earlier system state and they only have a transition-based history available to them. They must first have a mental image of either the current state or some earlier state. Then, using the transition-based history they must incrementally derive the previous or subsequent system state, depending on which mental image they start with, by updating that mental image until they reach the desired system state [Cowan & Wein, 1990]. Hence, a transition-based user history requires users to expend substantial mental processing effort in order to locate a system state so that they can examine some aspect of that system state. Thus, for this type of a task, the transition-based representation is not suitable when they need to access state information.

Specific Inform ation and Peripheral Inform ation

So far, we have concentrated on information *specific* to a user's interactions (e.g., user actions, system responses). However, history for reuse and history for error correction have also shed light on the need to maintain *peripheral information*. This is information, generated in the course of user interactions, that is not directly relevant but peripherally relevant to a user's immediate interactions. This information is important because it provides users with ''information for thought'' (i.e., information that can enhance both user interactions with a system as well as the user's understanding of their tasks and interactions). There are three types of peripheral information that should be maintained in a user history: *history commands*, *unaccepted user inputs*, and *snapshots*.

History commands are commands used to manipulate a history. They are considered peripheral to the immediate user interactions because they are not commands related to a user task but commands that expedite the specification of actions to perform a task (i.e., they allow users to reuse previous actions). No history tool maintains history commands, but they should because it may be advantageous for users to manipulate a history command rather than a task command (i.e., it may be faster and require less effort). Consider the scenario in Example 2 in which the next command to be issued can be derived by reusing either a history command or a task command.

history command		*task command*
		ls ~/thesis/talks
^talks^proposal	⇒	ls ~/thesis/ *proposal*
cd !-2:$	⇒	*cd ~/thesis/talks*
cd !-2:$	⇒	cd ~/thesis/ *proposal*

Example 2: A user intends to invoke the task command *cd ~/thesis/proposal*. It is a variation of an earlier task command which may be reused by typing a UNIX *csh* history command consisting of the word *cd* followed *!-2:$*. This history command retrieves the last parameter *~/thesis/proposal* from the second last task command (i.e., *ls ~/thesis/proposal*). However, it is the same history command used to generate the preceding task command, *cd ~/thesis/talks*. Another reuse operation, involving fewer keystrokes, is to invoke rather than retype the preceding history command by typing $!!^m$. $!!^m$ is a meta-history command which repeats the previous history command.

Unaccepted user inputs are user inputs that are rejected by a system because they are entered incorrectly (e.g., misspelled command) or used in an inappropriate context (e.g., trying to examine a file which is not in the current directory). Current history tools consider these inputs as errors and therefore, irrelevant. However, as suggested by the use of history for error correction, users may expedite error recovery by correcting and/or reusing the unaccepted user input (e.g., correcting the spelling of a command).

Snapshots are visual snapshots of a system state. This information is important because human memory research indicates that people do not attend to or remember all the information with which they are presented [Lindsay & Norman, 1977]. Users may have ignored the information by mistake or thought that it is not important for subsequent interactions. Therefore, while the snapshots may appear to capture extraneous information, they allow users to examine, in retrospect, aspects of those snapshots that they did not attend to earlier or remember. Furthermore, the snapshots contain visual contextual cues which can help users recognize information that is difficult to recall.

Externalized Information and Internalized Information

As users perform tasks using computers, they mentally retrieve information (i.e., *retrieved* information) that is relevant to the performance of the task. One useful aspect of history tools is that they cache this retrieved information so that users need not retrieve it from their memory when it is needed again.

Our discussion of options for history information has focused on retrieved information that has been communicated to the system (i.e., *externalized* information). However, history tools should not artificially restrict the retrieved information to only that which the users have explicitly communicated. That is, history tools should also capture *internalized* information; retrieved information that users do not communicate to the system during a session.

Since internalized information is not explicitly made available in the course of the interactions, users must either volunteer the information or the system must probe the user for it. In the latter case, Ericsson and Simon (1980) claim that there are only two kinds of information that a system can reliably probe for: a) information in short-term memory (i.e., *information in focal attention*) or b) information in long-term memory for which the system can provide direct probes. They claim that intermediate results of automated processing [1] or ongoing cognitive processes (e.g., recognition processes, perceptual-motor processes, and long-term memory direct retrievals) are unavailable, but final results are.

User and Behavioural Information

The use of history for user modelling and user interface adaptation point out the need to maintain, in a user history, information about individual users and their behaviours. Such information includes user preferences, aliases, task preferences, and user behaviours. They are inferred from an analysis of the user's interactions.

This type of history information depends on the kind of information being sensed and the technique for interpreting and inferring knowledge and behaviour from the sensed information. As alluded to in Chapter 2, both the sensed information and the inference techniques are many and varied. They need to be determined in advance and incorporated into history tools. As an illustration, Chapter 4 describes a behavioural phenomenon observed in user interactions – the repeated reference to a group of previously issued commands. Chapter 5 describes a technique for isolating this behavioural phenomenon. Such a technique should be incorporated in history tools in order to support this behavioural phenomenon.

Basic History Components

Different history uses require different history processing capabilities. For instance, history for reuse, inter-referential I/O, and error recovery require tools to reuse relevant history items, to relate them to current interactions, and to make corrections. As well, history for navigation and reminding require the display of the history for consultation purposes. Similarly, history tools require additional capabilities for exploiting particular types of history use. For instance, reuse of a previous interaction with possible modifications requires editing tools to alter the selected history item so that it can be reused properly in the current context. Reuse of a functional group of prior interactions requires tools for grouping items of a history so that they can be reused as group.

In order to support all seven history uses, our analysis of the requirements of history tools for each of the seven history uses reveal the need for minimally supporting the following four components: *collection*, *presentation*, *submission*, and

[1] Such behaviours have become fully automated and the associated knowledge and intermediate processings are integrated and composed into high-level procedures which are executed without being interpreted [Anderson, 1982].

administration [Barnes & Bovey, 1986; Joy, 1980; Lau & Asthana, 1984]. The collection component captures relevant information for a user history; the presentation component displays a user history; the submission component allows users to select history information for use in the context of the current situation; and the administration component manages, internally, a user history.

Collection of History

Caching relevant information helps users to conserve valuable internal memory stores but may incur cognitive and physical overhead costs [Schonpflug, 1988]. These costs include internally encoding aspects of history information and extraneous processing in order to access and locate information in the user history. Also, while the availability of vast storage capacities permit the retention of a great deal of history information, there is concern for the economic and efficient deployment of computational resources. Hence, judicious selection of history information is important as it influences the usability and effectiveness of history tools. Issues concerning what to collect, how much to collect, and what means are used to collect the information are addressed, in part, by two strategies: *selective logs* and *user collection tools*. Note that neither strategy is considered wholly effective or efficient.

Selective Logs

Selective logs automatically collect information concerning a user's recent interactions that is directly accessible from information exchanged during the interactions. Information that is indirectly accessible can also be collected using appropriate heuristics which infer and extract this information. The heuristics are based on empirical observations of user interactions and history use, or knowledge about task and interaction protocols. A case in point is Greenberg and Witten (1988b)'s study which observed considerable temporal bias in the commands that are repeated; on the average, 60% of the repeated command lines occurred within the last 10 command lines. Thus, they suggest that a simple strategy would be to keep the last 10 command lines issued as they are most likely to be repeated.

User Collection Tools

Designers also recognize the fact that some information is difficult to identify and access, and suggest the provision of appropriate capabilities that permit users to designate and to cull such information. *User collection tools* provide annotation capabilities which allow users to volunteer internalized information. The annotations can enhance the information obtained in *selective logs* or fill in missing information (see Example 3).

An alternative annotation scenario is to log all meaningful input and output information from user interactions. This log is then supplemented with tools that would allow users to cull the information when needed. An example is the facilities available in some window systems like SUNTOOLS that save substantial

> **Example 3:** A program management task involves a search subtask and program modification subtask. The search subtask involves scanning a hierarchical file system to locate source program files to be modified in the program modification subtask. Users may not be aware of the file search criteria. Therefore, they are unable to communicate this information as they sift through the file space looking for the desired files. An annotation tool would allow users to note the locations of the files so that they can be dealt with in the program modification subtask.

portions of old contents of windows. The contents can be reviewed with the aid of a scroll or search mechanism.

Presentation of History

The display of history is an important functional component of history tools. It augments user interactions by allowing users to access and use history information and provides an externalized representation of certain contents of a user's memory. As a clarification of terminology, the term *internal memories* refers to a user's own memories and the term *external memories* refers to any external media (e.g., notebook) which people use to record information from their internal memories. Two examples of external memories are visual stores and computer stores. Visual stores refer to a computer's display and computer stores refer to electronic files. The visual history mediates user access to external memories and facilitates certain mental and physical processing operations. Many of the current history tools, especially for reuse, are severely lacking in display support. Four different mental processing operations facilitated by a visual history and their associated support are examined:

- spontaneous reminding
- deliberate reminding
- visual scan and/or search
- problem solving and improvisations

Spontaneous Reminding

The old adage ''out of sight, out of mind'' is an apt introduction to a mental processing operation known as *reminding* which was first introduced in the discussion of history for reminding. A *spontaneous reminding* occurs when a user notices that an earlier situation shares certain similarities to the situation at hand [Ross, 1989]. Such phenomena may be nurtured by the presentation of a user history which provides *visual reminders of past experiences*. Also, by prominently displaying a user history, history tools help to augment a user's internal memories and to mitigate forgetting. Finally, the display of a user history leads to a user awareness of information that is not remembered or considered (i.e., it helps initiate reminding) as illustrated in the Example 4.

Example 4: A user has a vague recollection that a similar problem was previously encountered but does not remember the circumstances nor the solutions. In browsing a visual history, visual cues trigger the recall or recognition of relevant information and thereby help users to retrieve relevant knowledge about the problem and its solution.

Deliberate Reminding

When there are no visual reminders – from a lack of reminding information or visual representation schema – recall and recognition cannot be spontaneously facilitated. Then, the onus is on a user to initiate and pursue *deliberate reminding* by explicitly probing and extracting relevant history information. Such retrieval activities are fruitful if suitable cues are provided to elucidate the history information and to aid the retrieval of relevant history information.

There are cues that favour recognition mechanisms and those that favour recall mechanisms. The attributes that make cues effective are the subject of intense research and are particularly important if the potentials of history are to be realized [Larkin & Simon, 1987; Miyata & Norman, 1986; Reisberg, 1987; Schonpflug, 1988].

Visual Scan and/or Search

Recognition and recall operations may require that users perform a *visual scan and/or search* of their history. Many graphical systems exist today and they allow users to exploit the power of the display. A good graphical representation or diagram allows users to use quick perceptual judgements in place of abstractions and thereby expedite searches [Larkin, 1989]. Also, users may not have conscious access to scan and search knowledge, especially if perceptual knowledge is involved [Reisberg, 1987]. Finally, users may have difficulty in characterizing and communicating such knowledge because of limitations in interaction or communication capabilities. The display helps to bridge such difficulties by allowing a user to scan and search for the desired information.

Such visual processing operations are facilitated by a set of support capabilities. For example, users may need to locate and compare different pieces of history information which may be widely dispersed. A tool that allows users to locate and collect this information would be useful. Also, in order for users to identify and match a number of attributes, they need representational support to view and highlight information in different ways (e.g., fisheye views [Furnas, 1986]).

Problem Solving and Improvisations

Problem solving is often done in the context of an external display (Larkin, 1989 pg. 319). The display facilitates the process by reducing the complexity of the mental processes required. Furthermore, it is well-known that some people can think better with pictures because such thinking involves perceptual inferences as opposed to thinking with propositions which involves logical inferences

[Larkin, 1989]. As indicated in Chapter 1, *improvisations* occurring in a situation similar to one encountered earlier can benefit from access to history information from related situations. Also, a visual history can enhance problem solving. Thus, the display and a user history are possibly two of the many resources that users may use to support their problem solving or improvisations [Suchman, 1987].

Subm ission of History Item

One of the objectives of history tools is to reduce the physical effort required to specify commands after some form of the command has been previously externalized. The proposed benefits are increased speed, convenience, and economy of user actions as compared to that of retyping or repeating a user action. This type of support also frees the user from dealing with low level aspects of specifying commands or performing action sequences. The user is free to concentrate on the conceptual difficulties of a problem, especially when a similar problem has been previously encountered. This enables users to address difficulties associated with command specification and difficulties arising in the course of user interactions. In order to realize these benefits, the submission component must provide certain capabilities that will result in minimal distraction and minimal cognitive and physical effort. We propose four capabilities for facilitating submission of information from a user history: *identification* , *modification* , *retraction* , and *coordination* .

Identification

One important design requirement is to allow users to identify the desired history item for submission. There are two possible identification techniques: a) browsing a visual form of a user history to locate the history item and then pointing to it, or b) recalling the history item from a user history with a selection criteria. It is necessary to support both techniques as circumstances and individual preferences may call for the use of either technique. The concern is how to integrate the two techniques.

Selection of history information from a display is a natural use of the display. McMahon (1987) describes a mechanism which associates a type attribute to display objects and allows a particular interaction context to dictate the display objects that are selectable.

A history item is recalled when it is not visible, or it is difficult to locate visually, or it is more amenable to description than selection. The description may focus on certain distinguishing attributes of the item (e.g., a file name or a pattern occurring in the command) or on when the command occurred (e.g., last command). For example, in the UNIX *tcsh filename completion* mechanism, a user provides enough characters to identify a desired filename followed by an ESC character [Joy, 1980]. The facility completes the rest of the characters of the filename after consulting a list of filenames using the partial filename as a key.

Minimally, selection and recollection capabilities must be supported. A history type mechanism can be used to associate types to history information and to ensure that information appropriate to the current context can be selected or recalled. A history completion mechanism can be used to recall the particular history information from a list of candidates provided by the history type mechanism. A facility integrating such a history type mechanism and history completion mechanism supports the selection and/or recollection of the desired history information from a user history. Such a facility is considered in Chapter 6 where a number of history identification schemes are analyzed.

Modification

Information in a user history may be reused for some other purposes, may have been entered incorrectly and require correction, or may be an appropriate substitute for the desired information. Thus, since the selected history information is not always in a desirable form, users need tools to modify selected history information into an appropriate usage form.

There are three important issues. First, it is rare for users to select and edit history information in one step, partly because of the cognitive and physical effort involved in juggling both operations. Thus, it would be desirable to separate the two operations so that a user can verify that the desired item is correctly selected before changes are made to it. This separation also helps a user to refine the selection criteria if the wrong history information is selected.

Second, the modification capability should support features from a user's preferred editor so users need not learn new editor commands. Thus, existing editing skills can be carried over to history tool usage.

Finally, users may need to make modifications which may affect different elements of a piece of history information (e.g., words, arrangements of words, meanings, concepts, intentions). Thus, an editor must facilitate this. The question is how does the modification component provide such functionality and encourage users to explicitly indicate which elements in a user history are used? If users are encouraged to make explicit the elements that are to be modified in the process of making the modifications, then the modification facility could assist the collection component in annotating the information in a user history.

Retraction

The ability to retract operations is desirable for error recovery purposes. Also, retraction is a desirable function of a history tool itself because inherent limitations of history collection facilities, such as difficulties in detaching graphical actions from context, may result in the erroneous use of a user history. In association with a coordination tool, retraction permits user improvisation and error recovery to be initiated and coordinated. The retraction tool may be as sophisticated as Archer, Conway, and Schneider (1984)'s tree-based UNDO, REDO, and SKIP.

Coordination

Information in a user history may be submitted by either a) identifying and editing the information or b) composing user inputs using information from a user history (e.g., ls !$). The coordination capability is concerned with how history information is drawn into the course of user input composition. As suggested in the section entitled ''Contents of a User History'', there may be a number of different collections of a user history each maintaining different kinds of history information (e.g., locations of files, directories). In any usage situation, the desired information in a user history may come from one or more of these collections.

To support the formation of complex or compound commands, the coordination capability must allow users to group and generalize a number of history commands. The sophistication of this capability depends largely on the extent and the frequency with which users build complicated commands. The capability may be as sophisticated as a programming-by-example facility or as lightweight as a macro facility. However, it is reasonable to assume that the cognitive and physical effort involved in using the programming-by-example facility may limit a user's desire to use such a facility. Furthermore, we noted in Chapter 1 that user interactions are error-prone, improvisational, non-systematic in their recurrences, and interleaved with other activities. This suggests that users would prefer finer control than a macro facility would allow. Thus, users may opt for a facility that allows them to sequence and queue up a number of commands from a user history prior to submission.

Administration of History

The fourth history component is concerned with the internal organization and preservation of history information. There are three areas of concern:

- Separating independent user histories
- Accessing history information
- Aging, saving, and discarding history information

Separating Independent Histories

In many of the history tools examined, there is little or no logical separation of collections of a user history (i.e., into input and output or into task-specific groupings). In general, a user history is associated with a user's session. Such an ad hoc approach to maintaining a user history leads to a number of problems.

First, a user pursues a number of different activities concurrently within a session [Bannon, Cypher, Greenspan, & Monty, 1983; Cypher, 1986; Lee, 1992a; Miyata & Norman, 1986]. By grouping all the interleaved activities into a single collection, users may expend more time and effort to scan, search, and differentiate history information associated with a particular activity. As a result, they may be deterred by these inconveniences or they may be unable to capitalize on the benefits that the information in a user history provides. For example, users may not use their history to help them re-acquire the mental context for an activity because a great deal of effort is required to sift through their history to find the desired information.

It would be beneficial to maintain a user history appropriate to a particular set of activities, a work context or a task. Then, the issue becomes one of how users, in the course of their interactions, signify movement from one work context to another and thus associate subsequent activities to a particular work context or task. As one solution to this problem, a simple windowing mechanism allows users to logically organize different tasks into different windows where activities conducted in a particular window belong to one task. Shifts in user attention or switches in tasks may be explicitly and effortlessly conveyed to a system in the course of a user's interaction [Bannon, Cypher, Greenspan, & Monty, 1983; Cypher, 1986; Miyata & Norman, 1986].

Second, the artificial separation of history by session is problematic. The end of a computer session does not necessarily herald the completion of a set of tasks but instead, for example, some other engagement may preclude continuation at the present time [Bannon, Cypher, Greenspan, & Monty, 1983; Cypher, 1986]. Similarly, when users remotely log into a computer from another computer, the remote session is not part of the local session. Hence, there is now a second user history (i.e., one on the remote machine and one on the local machine) containing information that is not easily accessible on the local machine.

Finally, some history information spans more than one work context (i.e., it is global) and should be available to other work contexts as illustrated in Example 5. However, it is not clear how such information is extracted and placed into a global user history.

> **Example 5:** A UNIX user prefers to use the *ls* command with the *-F* option regardless of the particular user task. This command lists a directory's contents and marks items with an appropriate trailing symbol corresponding to the type of the item (e.g., a ''/'' for directories). Thus, this particular user preference should be kept in a global history rather than a user history for that task.

Accessing History Information

Occasionally, users may need to have access to history information that is older and/or not available on the display (i.e., it is in an electronic file). In such cases, they need capabilities that are akin to query and search capabilities which allow a subset of the history information to be extracted from the computer store for user perusal.

Aging, Saving and Discarding History Information

The final set of issues pertain to the management of history information in the display and computer stores. One display issue is how to *age* items in a user history so that items that are not referenced are less visible and less accessible. Also, the amount of screen real estate available to display a user history is limited and priority should be given to important history items. The issue of when and which items should be removed from the display is crucial because displacing important items may result in the loss of an item's service value to a user. These issues are reminiscent of the computer memory management issues for executing programs – a relationship which is explored further in Chapter 5.

One computer storage issue is how long to keep history information. Keeping information, that is never consulted or is ephemeral would needlessly consume system resources and would obscure more important information. On the other hand, some information *persists* for some time, possibly indefinitely, and premature removal would render a user history ineffective. This includes temporary information that concerns a particular active task or permanent information, like important or novel task solutions and user preferences or methods for adapting system behaviour (e.g., Example 5). Information concerning an active task should remain until the task is completed. Thus, the persistence of a history item – ephemeral, temporary, or permanent – needs to be identified and appropriate storage actions undertaken.

Concluding Remarks

The design of an integrated history tool must be concerned with two important questions: what information should be kept in a user history and what functionality should be provided to support user access to the information in a user history? With these two questions in mind, our analysis of the seven uses of history tools led us to propose several answers (Table 1 summarizes our proposals).

While the seven uses of history can provide one possible source of insight to these two questions, additional corroboration and elaboration may be available from other sources. One source is naturalistic observations of user interactions which provide insights into behavioural characteristics and patterns evident in

Question	Options	Examples
Contents of User's History	Actions and objects of actions	full commands, file names, and directories
	System responses	selected state information
	States and transitions	snapshots of states and user actions
	Peripheral information	history commands, unaccepted user inputs, snapshots
	Internalized information	information in short-term memory and information in long-term memory that can be probed
	User and behavioural information	user preferences and behavioural phenomena
Basic History Components	Collection	selective logs, user collection tools
	Presentation	spontaneous and deliberate reminding, visual scan, and/or search, other mental processing operations
	Submission	identification, modification, retraction, coordination
	Administration	separate collections of a history, tools for internally managing a history

Table 1: Questions, options, and examples examined.

user interactions. These behavioural insights are the bases for heuristic techniques used by history tools to capture relevant information from user interactions. The studies in Chapter 5 are rigorous examinations of a behavioural phenomenon observed in a naturalistic study of UNIX user interactions.

Relevant history information may also be based on a theory of user interactions. For instance, Uejio, Blattner, Schultz, and Ishikawa (1991) describe two history tools developed from two different models of command retrieval. One command retrieval model suggests that users search for commands based on some semantic relationships, concepts, attributes, and abstractions. The corresponding history tool logs complex commands and maintains various attributes about these commands: frequency of use, files accessed, time executed, and keyword descriptions. The other command retrieval model suggests that, sometimes, users have a fuzzy recollection of the information. A neural network is used to maintain previous user commands and to allow users to recall particular history information using an incomplete specification of the desired information.

While behavioural studies and theories of user interactions can provide insights concerning useful history information and capabilities, they do not substitute for insights from a detailed study of user behaviour. Studying how users use prototype history tools or asking users about what history information and history functions they would like can be equally informative.