A Taxonomy of Uses of Interaction History

Alison Lee

Department of Computer Science
University of Toronto
10 Kings College Road
Toronto, Ontario
M5S 1A4
alee@db.toronto.edu

Abstract

A variety of tools have been proposed to enhance and support user-computer interactions [Lee 1987]. One such tool is the *interaction history facility*. It permits the user to have access to *past interactions* kept in a *history* and to incorporate them into the context of the current situation. We characterize different ways the history can aid in the performance of a user's tasks. The list of possible aids include: history for reuse, history for recording & replaying a script, history for user recovery, history for navigation, history for external memory support, history for adaptive interfaces, and history for user modelling. We conclude with a discussion of some of the issues and problems that this taxonomy has helped to raise.

Résumé

Une variété d'outils ont été proposés dans le but d'améliorer et de supporter l'interaction d'un usager avec l'ordinateur [Lee 1987]. L' *interaction history facility* est l'un de ces outils. Il permet à l'usager d'avoir accès aux *interactions antérieures* conservées sous le nom d'histoire (*history*) et de les incorporer au contexte de la situation courante. Nous décrivons pour les usagers différentes façons d'utiliser l'histoire de manière à améliorer la performance de leurs tâches. Parmi la liste des utilitaires, on retrouve: l'histoire pour la réutilisation, l'histoire pour enregister et réexécuter un scripte, l'histoire pour la récupération-usager, l'histoire pour le pilotage, l'histoire pour le support de mémoire externe, l'histoire pour les interfaces adaptives, et l'histoire pour la modélisation-usager. Nous concluons avec une discussion portant sur les questions et problèmes que cette taxonomie aura permi de soulever.

Keywords: user support, history, script, reuse, inter-referential I/O, user recovery, macros, programming with example, navigation, external memory support, adaptive interfaces, and user modelling.

1. Introduction

Numerous run-time, computer-based support tools have been proposed to assist the user's immediate and ongoing interactions with the system (i.e., *user support tools* [Lee 1987]). By and large, the HELP tool has garnered most of the attention judging by the literature on this subject. However, there is a scarcity of information about *interaction history tools* (history tools for short).

These tools permit users to access *past interactions* kept in a *history* and to incorporate them into the context of the current situation. The basic history tool is made up of four components, minimally: *collection*, *presentation*, *selection/modification*, and *submission* [Barnes and Bovey 1986, Joy 1980, Lau and Asthana 1984]. The *collection* component records past user-computer interactions into a *history*. The *presentation* component displays the history. The *selection/modification* component allows the user to copy (and possibly modify) a history item. The *submission* component allows the user to *use* the selected history item in the context of the current situation.

History tools are considered because of their potential as a user support tool. Such a tool does not need to be highly knowledgeable about the task or be application specific or act autonomously in order to be of assistance to the user. It would exploit the computer's strengths (e.g., storage and search) to compensate for human limitations (e.g., memory and focus of attention), to allow users to concentrate on the conceptual rather than the tedious and mundane aspects of the task, and to deal with problems that crop up in the course of their interactions. Essentially, it is an *electronic assistant* who provides users with extra hands and eyes. As we shall see, the history tool fits this bill by providing a number of aids that are supportive of user-computer interactions.

In section 2, we introduce the taxonomy. In section 3, we examine the individual uses enumerated in the taxonomy. We conclude with a discussion of some of the issues and problems that this taxonomy has helped to raise.

2. Taxonomy

The taxonomy identifies seven different uses of the history (see Table 1). The list grew out of our survey of current systems that support the history tool by name as well as those that maintain some form of a *history* that users may use directly or indirectly. We cast our coverage widely and completely with respect to the possible user aids that current history tools provide. Of course, there are many other uses that fall outside of our scope of interest (e.g., using the history of user's session to analyze the design of a system).

Uses of History	Types of Uses	Initiated By User	Initiated By System	Repository Actions	Repository Data
reuse	repeat an operation	√		√	
	repeat as a functional group	√	(√)	√	
	relate input and/or output	√		√	√
recording & replaying a script		√		√	
user recovery		√	(√)	√	
navigation	information spaces	√			√
	activity spaces	√		√	
external memory	consult	√		√	√
	remember	√		√	√
adaptive interfaces			√		√
user modelling			√		√

Table 1: Taxonomy of uses of interaction history.

Each use is distinctive and reflects the user's basic intention for the history use. However, within a particular use, there can be varying *types* of uses and these can overlap. They reflect the choices that are available in carrying out the basic intentions unconstrained by the system or the situation. To illustrate, let us consider *repeat an operation* and *functional grouping*. The former reuses a previous command or object and the latter reuses a group of commands as a unit. The differentiation is made because some systems support this capability directly while others support it with difficulty, if at all. More importantly, when the user has the intention to do a functional grouping and the system does not directly support or match the user's intention, then the level of interaction required by the user is more primitive and the amount of user involvement to realize the goal is greater.

We distinguish amongst some of the different types of uses with respect to special features and implementation characteristics; nominally referred to as the *forms* of history use. The different *forms* and *types* of history uses along with some example systems that support them are listed in Table 2.

Each history use can be *initiated* by the *user* or by the *system*. All but history for adaptive interfaces, user modelling, and in some cases user recovery and reuse are in the user-initiated category. History uses that fall in the system-initiated category are able to help the user by predicting and initiating actions. While they have a strong appeal, their realizations are generally domain-specific and limited in scope.

The contents of the history can be *viewed* as a *data* or *actions repository*. In the first case, the contents of the history are referenced by the the user or the system to assist their processing (e.g., user's preferences). In the second case, the contents of the history are invoked by the user or the system as an action to be performed (e.g., repeat an action, undo an action).

There have been two previous surveys of history. Greenberg 1988 examined different interaction styles for implementing reuse based on history. Linxi and Habermann 1986 examined various ways of keeping and reusing history in the context of improving the software development process. Our taxonomy examines history as a user support tool and elaborates on uses beyond reuse.

3. Uses of Interaction History

Herein, we identify and examine seven uses of history. The terms *history*, *scripts*, and the *future part of the history* appear to be used interchangeably. To clarify, a *history* is a log of information and actions that have taken place sometime earlier. A *script* is a sequence of actions to be carried out which may incorporate temporal elements to coordinate activities and to specify transformations on objects [Archer Jr. et al. 1984]. If during the recording, the actions are performed for real, then the recording (aside from being the script) is part of the history. A future instantiation of the script could be a *future part of a history*. Viewing the user's interactions with respect to an activity timeline, actions and objects that took part earlier are history and thus unmodifiable (as it reflects the historical log of the user's session), and actions and objects that are to take place in the future as either a part of a script or as a future action are thus modifiable up to and just prior to their taking place.

3.1. History for Reuse

Currently, the most common use of history is to reuse history items (possibly with modifications) to save keystrokes or mouse strokes [Linxi and Habermann 1986]. However, there are a number of choices available to the user.

Repeat an Operation

This permits the user to repeat a single operation (typically a command line) by one of two means. The forms arise because of the different emphases placed on how history items are selected and/or modified.

Systems that model interactions as *typescripts* (i.e., transcript of input and output) maintain histories of the command lines issued. Items are selected by *descriptive/indirect manipulation* or by *direct manipulation*[1]. The former requires the user to remember the history items and syntactical constructs (e.g., !-5:s/aa/bb recalls the 5th last command substituting the first occurrence of *aa* by *bb*) and to juggle the two as the history item is copied and edited. In the latter case, items are visible and directly manipulable by the users with control keys or a mouse (see Figure 1 for a sample display of the INTERLISP-D HISTMENU). There are situations in which one style of selection is suitable over the other. For instance, descriptive manipulation is suitable when references to a particular history item can be described easily and uniquely by a pattern. On the other hand, there are situations where it may be more expedient for the users to scan and point to the item of interest. Furthermore, the visual presentation and immedi-

[1] Greenberg and Witten 1988 calls these, respectively, *history through glass teletypes* and *history through graphical selection*.

Uses of History	Types	Forms	Example Systems
reuse	repeat an operation	command typescripts by	
		descriptive manipulation	C Shell [Joy 1980], COUSIN-UNIX™ [Hayes and Szekely 1983], INTERLISP-D USE, REDO [Teitelman and Masinter 1986]
		direct manipulation	KORN Shell [Korn 1983], HCR HI [HCR Corporation 1987], Edit Shell [Steffen and Veach 1983], MINIT [Barnes and Bovey 1986], TC Shell [Ellis et al. 1987], INTERLISP-D HISTMENU & FIX [Teitelman and Masinter 1986]
		browsing/editing	
		into scratch area	CEDAR [Teitelman 1984]
		at input focus	CEDAR [Teitelman 1984], Macintosh UW [Bruner 1985], SUNTOOLS [SUN Microsystems, Inc. 1986]
		at the command line	EMACS [Stallman 1981], CEDAR [Teitelman 1984]
		to the window workspace	CEDAR [Teitelman 1984], SMALLTALK-80
	relate input and/or output	input to input	SYMBOLICS [McMahon 1987]
		output to input	SYMBOLICS [McMahon 1987]
	repeat as a functional group	macros	
		history macros	ALOE [Linxi and Habermann 1986]
		recorded macros	EMACS [Stallman 1981], HP NEWWAVE AGENT [Stearns 1989], MACROS BY EXAMPLE [Olsen and Dance 1988], MEXEC [Ash 1981], TEMPO [Whitby 1986], QUICKEYS [Bobker 1988]
		programming by examples	METAMOUSE [Maulsby et al. 1989]
		programming with examples	SMALLSTAR [Halbert 1984], PERIDOT [Myers and Buxton 1986]
recording & replaying a script			ACTIVE PATHS [Zellweger 1988], CONMAN [Haeberli 1986], MIT Lincoln Labs [Lancaster-Thomas 1969], PLAYERPIANO [Bier and Freedman 1985]
user recovery		history-based UNDO/UNDO	INTERLISP-D UNDO [Teitelman and Masinter 1986]
		linear UNDO/REDO	CHIMERA [Kurlander and Feiner 1988]
navigation	information spaces	data retrieval domain	WHAT, WHERE, WHENCE [Engel et al. 1983]
		hypertext domain	HYPERCARD RECENT [Goodman 1987]
		documentation domain	timeline page [Feiner et al. 1982]
	activity workspaces	list of active jobs	JOBS in C Shell [Joy 1980]
		record of user's excursions	SITES, MODES, and TRAILS [Nievergelt and Weydert 1980], Room Stack in ROOM MODEL [Chan 1984]
		the previous activity	Back Door in ROOMS [Card and Henderson 1987]
external memory support	consult	progress of active jobs	[SUN Microsystems, Inc. 1986, Teitelman and Masinter 1986, Teitelman 1977]
		re-orientation	[Engel et al. 1983, Nievergelt and Weydert 1980]
		track and debug errors	
	remember		reacquire mental task context
			guide in performing similar task
adaptive interfaces			default menu selection to last
			REACTIVE KEYBOARD [Witten et al. 1983]
user modelling			UNIX™ CONSULTANT [Chin 1986], UKNOW [Desmarais and Pavel 1987], STEREOTYPE [Rich 1983], and [Senay 1989]

Table 2: Uses for interaction history tools and examples

ate feedback enhances the the quality and support for selection and modifications of history items and provides an external reminder to the user of the contents of history.

History for reuse is available in systems that allow users to copy text appearing anywhere on the screen to a work area (see Table 2 and Figure 2) where further editing may be performed before it is submitted[2]. In these systems, browsing and editing, coordinated through a common representation of the application objects, is *the* interaction paradigm [Scofield 1981, Young et al. 1988]. Here, the common representation is the textual contents of the screen which is also the history. Unlike command typescripts, this form of reuse does not require an explicit history support machinery to be built (i.e., available virtually for free). No overhead is incurred for the collection and presentation of the history as the screen is the history repository and, in effect, the history is always on display. By having the history constantly on display, browsing rather than querying becomes the primary means of examining the history. Furthermore, history items are not limited to the command lines entered but include the results or outputs from executing the command. As well, editing is an integral part of the interaction paradigm so that support for modification of a history item is provided with little or no effort. Overall, selection and modification of a history item in this approach is manual but direct, performed at a lower conceptual level, and much improved compared to the first approach (i.e., command typescripts).

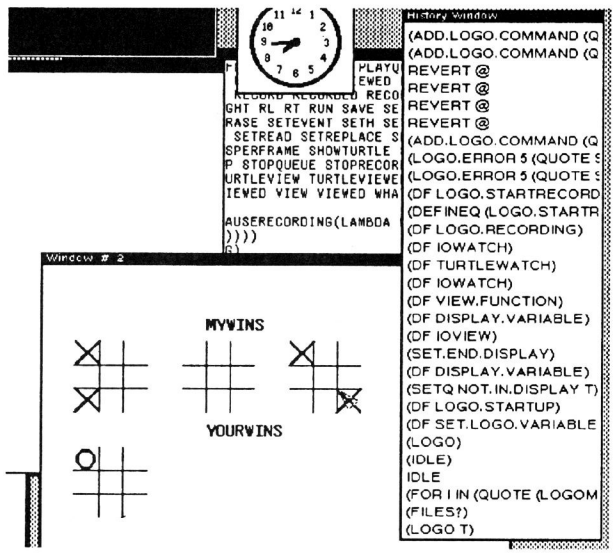


Figure 1: The INTERLISP-D HISTMENU package displays the history of the commands issued to the Executive in a menu. The user may select the items from the menu (the window entitled *History Window*).

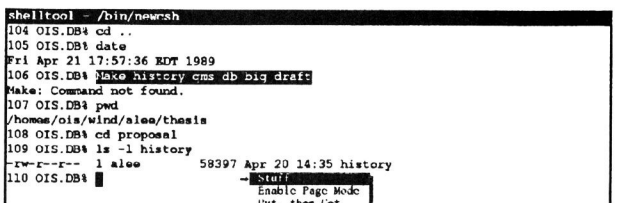


Figure 2: In SUNTOOLS, text may be selected, copied, and stuffed to where the input focus is (i.e., ▮).

Relate Input and/or Output

In human-human dialogue, conversants can make abbreviated references to objects and actions that took place in the earlier dialogue without either of them having to repeat it in part or in its entirety [Draper 1986, Reichman-Adar 1986]. In the computing analogue, the user's input or the system's output in the current dialogue (available on the display) can reference previous input or output or both (called *inter-referential I/O* [Draper 1986]).

History is an explicit component of this capability because the input and/or output that the conversants use originates from an earlier part of the dialogue. Beyond simply reusing the input and/or output, the user and computer converse in a manner that makes explicit the relationships between the actions and objects of an earlier part of the dialogue with the current part of the dialogue. Hence objects and actions are explicitly disambiguated, albeit in a more abbreviated form. However, this does not fully realize the capability that exists in human-human dialogue as we are nowhere close to achieving the understanding that human conversants are able to of each other's mind. In human-human dialogue, conversants have access to other conversational cues (e.g., body language, intonation, facial expressions) that help disambiguate the dialogue. These are not available in current human-computer dialogue but the shared display and explicit denotation of object and actions in the dialogue is a positive step towards achieving understanding between conversants.

In the SYMBOLICS GENERA programming environment [McMahon 1987], users make references to objects or representations of actions on the display by pointing at them. The underlying mechanism that makes inter-referential I/O possible is a type mechanism that associates types to *user interface data* (known as *presentation types*). Conceptually, the presentation type relates the piece of data with the way it is to be used in particular user interface situations. These presentation types are arranged in an inheritance lattice that organizes the way in which user interface data are collected and returned to the user. An example is when a print request requires operands (i.e., filename). Objects on the display that have the desired presentation type (i.e., filename) are mouse sensitive [McMahon 1987].

History for Functional Grouping

In this type of reuse, users group a set of items into one functional unit so as to construct a compound/complex command. They come in three forms: *macros*, *programming with example*, and *programming by example*.

[2] Greenberg and Witten 1988 calls it *history through editing*.

A macro facility [Ash 1981] permits the user to associate one instruction, command, keystroke, or mouse action to a sequence of instructions, commands, keystrokes or mouse actions. To reflect how the macros are constructed, there are *history macros* and *recorded macros*. *History macros* are constructed by selecting pieces of the history and binding them as a macro. With *recorded macros*, the user enters a record mode and the commands issued subsequently define the macro. By and large, macro facilities are limited. In a graphics-based system, a user is unable to view and edit the resulting macro. Most systems do not support parameters and control constructs in the macros.

When the system can make inferences and generalizations from the examples the user gives, they are known as *programming by example* systems (e.g., METAMOUSE can induce graphical procedures from the user execution traces of a drawing program [Maulsby et al. 1989]). Few systems are able to make inferences and generalizations [Myers 1986].

Typically, a user demonstrates to the system an example of a procedure and then proceeds to indicate how to generalize the resulting trace using an editor. The resulting program or function is more sophisticated and is meant to perform a single complete task. The history is augmented with logic, symbolic computations, and object descriptions to capture and encapsulate all the objects and operations for the specific task. Such facilities are called *programming with example*[4]. Figure 3 illustrates an example construction of a SMALLSTAR program.

3.2. History for Recording & Replaying a Script

The oldest and most primitive use on our list allows the user to record and replay a sequence of actions verbatim, effectively "pushing the buttons" for the user like a player piano. The recorded script can be used for canned demos, as a debugging test suite, for performance benchmarking, for configuring systems, and for distributing new releases.

By and large, script editing is unsupported by the tool itself. Typically, such systems only maintain information necessary to replay the script. Thus, to reuse the script for purposes other than what it was intended to do would require substantial tweaking on the user's part. The onus is on the user to understand the script (which is not necessarily in a human-readable form), to know what modifications to make (using some editor), and to perform them correctly. It is intended to perform a specific script and this is why it is not considered a variation of reuse.

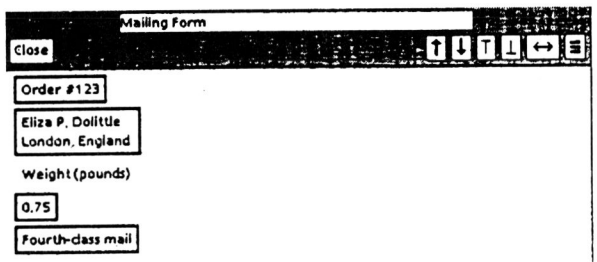
3.3. History for User Recovery

User recovery tools allow users to recover from unforseen errors (e.g., typing) and to experiment with the system's advanced commands. Typically, user recovery is facilitated by UNDO, spelling correction, and editing. UNDO is pertinent to our discussions. History based implementations of this operation exists as do others.

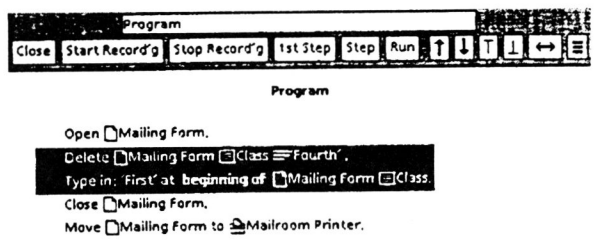
[3] Linxi and Habermann 1986 calls the latter *keyboard macros* to reflect the fact that in some systems (e.g., EMACS, Tempo), the macro may be bound to one key for quick easy invocation.

[4] Greenberg and Witten 1988 does not distinguish the two.

In the mail form, the **Weight** field is examined to determine what should be filled in the **Class** field.



The initial program after the user records the actions assuming that the value in the **Weight** field is less than 1 pound. The user selects the statements that will be in the body of the conditional to be provided.



This shows the completed program with the correct conditional test and actions.

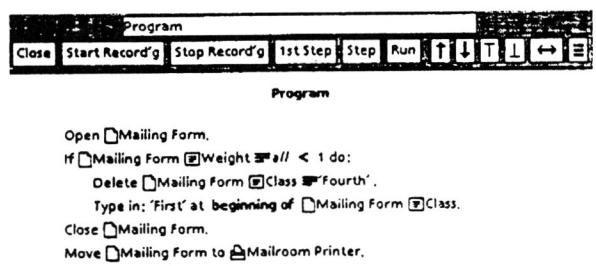


Figure 3: The SMALLSTAR program determines whether a customer's order is to be sent by first-class or fourth-class mail. If the value in **Weight** field of the mail is less than one pound, the order will be sent by first-class mail, otherwise by fourth-class mail.

According to Vitter 1984, *history-based UNDO/UNDO* maintains a history of all the primitive and recovery commands (i.e., UNDO) ever executed[5]. Users may use UNDO to undo the effects of one or more previous commands. However, to UNDO the nth earlier command, the intervening n-1 commands must be undone first. Issuing an UNDO command on an UNDO reverses the effects of the first UNDO. The power of this facility is that it can return the system state to any value that

[5] We do not make a distinction about how the operation is internally realized (e.g., restoring to an earlier snapshot or state of the system or by reversing changes at each step).

existed earlier but not one that did not exist formerly. That is, the user cannot backtrack to some state, invoke a couple of new operations and then carry on with the operations following the point from which he backtracked to (referred to as command insertion by [Vitter 1984]). This implementation constraint has a direct impact on whether the user's intention can be realized (i.e., when some commands cannot be undone).

In *linear UNDO/REDO*, the UNDO and REDO commands are meta-commands which act on primitive commands (i.e., UNDO/REDO cannot operate on other UNDO/REDO). This form of user recovery allows users to insert operations into the midst of the history (which the pure history-based UNDO/UNDO cannot handle). However, if the user has a change of heart and does not really want to insert those operations in the chain, then they cannot be removed (which the pure history-based UNDO/UNDO can handle) [Vitter 1984]. An example of a system based on this model is the graphical history facility in CHIMERA [Kurlander and Feiner 1988].

3.4. History for Navigation

History for navigation allows the user to reflect on where they have been and where they are, and to use that information to guide their progress [Engel et al. 1983, Fitter 1979, Nievergelt and Weydert 1980, Paap and Roske-Hofstrand 1988]. Navigation is possible in one of two spaces: *information spaces* and *activity workspaces*. The history is generally presented to the users as a series of static frames of places that they visited. These static frames can potentially contain a wealth of information (e.g., actions performed, progress of actions, errors encountered) and are generally presented in one form (e.g., temporal order). However, when the frames were visited is not necessarily the appropriate presentation when they want to locate only frames that pertain to an activity (i.e., activity structure may highlight the desired information better).

Information Spaces

This is a common type of navigation in the information retrieval domain; other example domains are listed in Table 2. Users can easily become lost in the vast information space as they navigate through it. The history as a navigation aid can help minimize this. It can provide information about where users are presently, where users last visited, and where they have visited. Figure 4 shows the *timeline* page of [Feiner et al. 1982] which shows the pages of a document that the user has seen in miniature form ordered chronologically along bands associated with their parent chapters.

Activity Workspaces

This type of navigation allows users to move freely back and forth between and within activity workspaces [Bannon et al. 1983, Card and Henderson 1987, Cypher 1986, Lee 1987, Miyata and Norman 1986]. Such facilities maintain a history of job activities in various forms. In one form, users can use the display information to find out about the progress of current and other activities. The information is directly accessible when separate activities are performed in separate windows (e.g., windowing system). In another form, the user may find out what the active jobs are by an explicit command like JOBS in UNIX™ C Shell or by examining the display to see

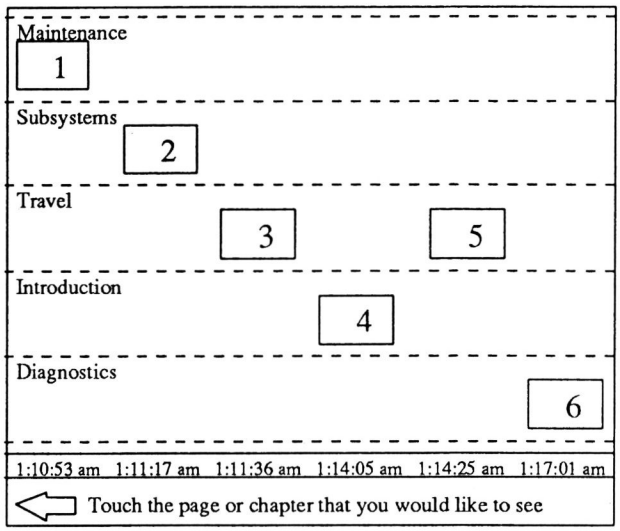


Figure 4: Sketch of the *timeline* page from [Feiner et al. 1982]. The last 6 pages that were viewed are displayed chronologically in miniature form along the band associated with its parent chapter. Selecting the miniature will return the user to that page. Other pages in the timeline are displayed by selecting the scroll arrow.

what is incomplete. The user may also find out the chronological order of activities he is working on (e.g., the Room Stack in ROOM MODEL [Chan 1984]). A primitive form of this is the user's last activity (e.g., Back Door in ROOMS [Card and Henderson 1987]).

3.5. History as External Memory Support

Users can *consult* the history or use it to help them *remember* information associated with past activities. This use of history is apparent in any system that displays a history of the user's session. The displayed history is an external memory aid which the user can reference at appropriate junctions in their interactions. However, the onus is on the user to extract and interpret the information in the history (i.e., perform most of the processing associated with using this history use). This can potentially be an invaluable history use but the effort generally required of the user limits the extent to which users exploit the information contained in the history. Thus only the easily accessible information are utilized while other information are accessed when the value outweighs the effort to access it. As we alluded to in the navigation case, if the user can query or reorganize the presentation of the information (e.g., hiding irrelevant information or details) both consultation and reminders would be more extensively utilized. The lack of such capabilities can severely limit the value of history as an external memory support tool.

In the consultation case, users use the history to help them answer questions and queries they have before deciding what to do next. For instance, users could find out what activities are incomplete and require attention (commonly when the user is engaged in a number of activities in a parallel or

interleaved fashion). In another example, the history helps them to figure out how and where they may have introduced an error(s) in the course of their interactions. Finally, users have questions about how to do a task and the history can contain some solutions (albeit not necessarily optimal but sufficient for the users' needs) which they explored previously (in full or in part) or which are appropriate to the current task.

In the second case, users who return to a previously suspended activity after a digression can use the history to help them to reacquire the mental context for the activity. User may lose track of what they are doing because of an interruption and the history can remind and re-orient them. Also, users may simply recall having performed a task similar to the current one and they use the history for that task as a guide for completing the current one.

3.6. History for Adaptive Interfaces

The history is a source of information which the system can use to automatically adapt the interface behavior to suit the user's needs. Basically, the system uses heuristics to predict what users might want to do next or what their preferences are. Greenberg and Witten 1988 calls this *history through prediction* whereby the system estimates for each token already seen the probability that it will be the next one typed. The entries with the highest probabilities are made available for selection. A simple-minded example is a menu system that makes the last selection be the default selection by either relocating that item to the top or positioning the cursor to that item when the menu appears. Another example is the REACTIVE KEYBOARD system [Witten et al. 1983]. Based on text previously entered by the user, the system computes a probability for each character that it will be the next one to be typed and offers it up to the user when text input is required from the user.

3.7. History for User Modelling

The history is also a valuable information repository for implicit user modellers to infer or derive information about the user [Chin 1986, Desmarais and Pavel 1987, Rich 1983, Senay 1989, Tyler and Treu 1986]. Basically, implicit user modellers monitor or observe what the user does, how the user uses the system, etc. to formulate user models. User modellers can determine the user's skill level, command and task knowledge, preferences, and personality traits (see Figure 5 for an example) [Lee 1987]. The techniques that systems use to draw the information include deterministic, probabilistic, behavior to structure transformations, induction, and knowledge inference.

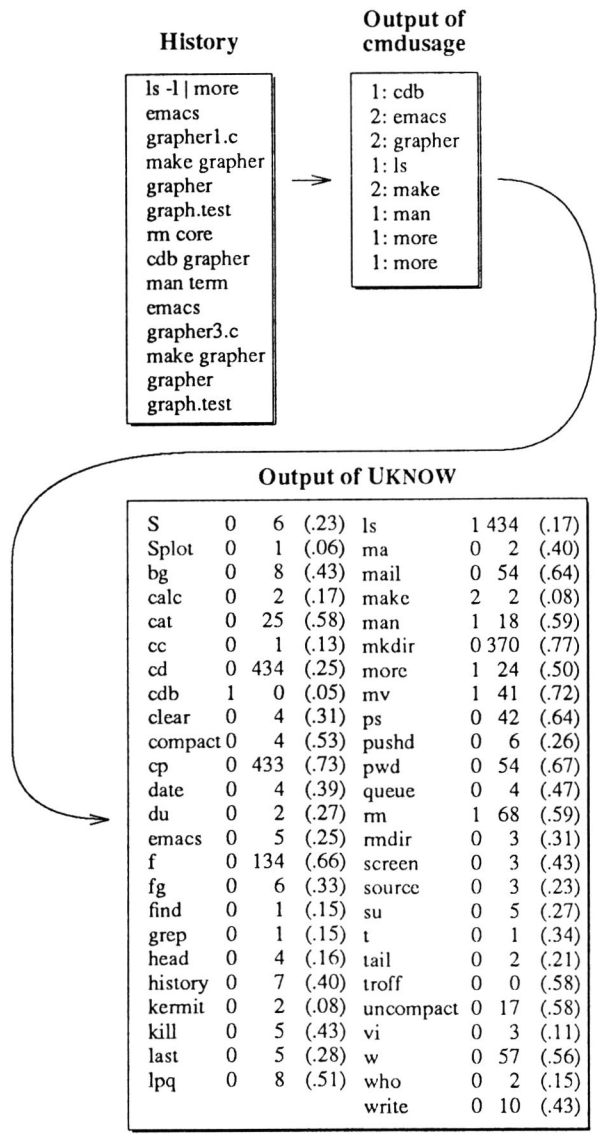


Figure 5: 'cmdusage' analyzes the user's history. UKNOW [Desmarais and Pavel 1987] uses this information to identify the extent of the user's knowledge of of the commands.

4. Open Problems and Issues

The taxonomy identifies a wide range of different uses that an interaction history can provide. It has brought to light a number of open issues and problems: scarcity of information about interaction patterns and history usage characteristics, basic design concerns, and architectural concerns associated with history as a user support tool [Lee 1989a].

As history tool designers, we are interested in the behavioural data that exist concerning the nature of human-computer interactions and history usage characteristics (i.e., nature, frequencies, and sophistication). There have been studies but they concentrate on command use. Findings characterizing the nature of human-computer interactions can give us insights into interaction patterns that would be relevant to any user support efforts [Bannon et al. 1983, Greenberg and Witten 1988, Henderson and Card 1986, Hanson et al. 1984, Lee and Lochovsky 1990]. Studies of users using history tools would allow us to investigate and identify *the potential and actual uses of history tools*, inadequacies in systems, the prevalent uses, and uses not enumerated in our taxonomy. Furthermore, there have been no studies examining the cognitive and physical effort associated with using particular designs of history tools. This can give insights to the issue of whether such effort outweigh the benefits the various uses of history offer (i.e., excessive effort deters use?).

Two studies recently have demonstrated that repetition of individual operations (74% on average) and groups of operations exist in user's interactions [Greenberg and Witten 1988, Lee and Lochovsky 1990]. These results have direct implications for history for reuse. The Greenberg and Witten study found that actual uses of the C Shell history tool were minimal. Our exploratory study of two history tools (C Shell and TC Shell) [Lee 1988b] corroborate their findings. Furthermore, our qualitative analysis shows that history usages are unsophisticated with a large majority of the history commands being confined to simple specialized operations (e.g., !!). Both studies are limited. In particular, the Greenberg and Witten study only looked at reuse and did not investigate any of the other possible uses. Our study was extremely exploratory and qualitative.

Aside from the behavioural and usability issue, there are also a number of unresolved design issues [Lee 1989a]. First, what should the *history* include? We propose a number of possibilities: input as well as output objects, textual and non-text objects, and less directly accessible information like user goals, intentions, and tasks. Second, which of the history information can be obtained directly and automatically and which must be provided with the user's own input? In the latter case, what techniques may be used to obtain the information? Third, what portions of the *history* should be displayed (permanently and on demand) and what techniques (e.g., signals and descriptions, overlapping windows, switching between various display panels) and representational schemas (e.g., icons and fisheye views) can be used for reminders, external memory support, and cues for recognition and recall. Fourth, what level of support and degree of functionality should be provided for selection and modification of a selected history item ? Fifth, how should the history be organized (by function or by task), which items need to be integrated and which items need to be kept separately, and what support is needed to manage the *history*? (e.g., accessing other histories, querying history, aging and discarding history)? Finally, which aspects of the history tool are amenable to mechanization?

The last major issue pertains to the architectural support that must be provided for the operation of history tools. The framework is needed to ensure that the history information from the relevant level be collected. Second, history concepts and functionality must pervade throughout the system and transcend particular applications (i.e., history is integrated and available from all applications). Third, the system should be flexible to support new concepts, uses, and capabilities. There are very few efforts that are specifically concerned with identifying and providing coupling architecture or framework to integrate user support facilities [Cockton 1989].

Acknowledgements

Special thanks to Caroline Houle for providing the french translation of the abstract. This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant OGP0003356.

References

J.E. Archer Jr., R. Conway, and F.B. Schneider (January 1984). User recovery and reversal in interactive systems. *ACM Transactions on Programming Languages and Systems*, **6(1)**, pages 1-19.

W.L. Ash (August 1981). Mxec: Parallel processing with an advanced macro facility. *Communications of the ACM*, **24(8)**, pages 502-509.

L. Bannon, A. Cypher, S. Greenspan, and M.L. Monty (Dec. 12-15, 1983). Evaluation and analysis of users' activity organization. In *Proceedings of the CHI'83 Human Factors in Computer Systems*, pages 54-57, Boston, Massachusetts.

D.J. Barnes and J.D. Bovey (September 1986). Managing command submission in a multiple-window environment. *Software Engineering Journal*, **1(5)**, pages 177-184.

E. Bier and M. Freedman (1985). PlayerPiano: Playback scripts for window systems. Unpublished paper, 22 pages.

S. Bobker (January 1988). Command performance. *MacUser*, **4(1)**, pages 114-122.

J.D. Bruner (November 8, 1985). UW: A multiple-window terminal emulator for use with 4.2BSD UNIX. Manual for version 2.1 of UW, 14 pages.

S.K. Card and D.A. Henderson Jr. (April 5-9, 1987). A multiple, virtual-workspace interface to support user task switching. In *Proceedings of the CHI+GI'87 Human Factors in Computing Systems and Graphics Interface*, pages 53-59, Toronto, Ontario.

P.P. Chan (July 1984). Learning Considerations in User Interface Design: The Room Model. Technical Report CS-84-16, 51 pages. Computer Science Department, University of Waterloo.

D.N. Chin (April 13-17, 1986). User modeling in UC, the UNIX Consultant. In *Proceedings of the CHI'86 Human Factors in Computer Systems*, pages 24-28, Boston, Massachusetts..

G. Cockton (August 21-25, 1989). Session discussions. In *Proceedings of the Working Conference on User Interfaces (Engineering for Human-Computer Interaction)*, Napa Valley, CA.

A. Cypher (1986). The structure of users' activities. In D.A. Norman and S.W. Draper (editors), *User-Centered System Design: New Perspectives on Human-Computer Interaction*, pages 243-263. Lawrence Erlbaum Associates.

M.C. Desmarais and M. Pavel (September 1-4, 1987). User knowledge evaluation: An experiment with UNIX. In H.J. Bullinger and B. Shackel (editors), *Human-Computer Interaction - Interact'87. Proceedings of the Second IFIP Conference on Human-Computer Interaction*, pages 151-156. North-Holland (Elsevier Science Publishers B.V.).

S.W. Draper (1986). Display managers as the basis for user-machine communication. In D.A. Norman and S.W. Draper (editors), *User-Centered System Design: New Perspectives on Human-Computer Interaction*, pages 339-352. Lawrence Erlbaum Associates.

M. Ellis, K. Greer, P. Placeway, and R. Zachariassen (March 10, 1987). TCSH - C shell with filename completion and command line editing. UNIX Programmer's Manual (revised U of T, previous revisions from Fairchild, HP Labs., and OSU IRCC) .

F.L. Engel, J.J. Andriessen, and H.J.R. Schmitz (1983). What, where and whence: Means for improving electornic

data access. *International Journal of Man-Machine Studies*, **18(2)**, pages 145-160.

S. Feiner, S. Nagy, and A. van Dam (January 1982). An experimental system for creating and presenting interactive graphical documents. *ACM Transactions on Graphics*, **1(1)**, pages 59-77.

M. Fitter (May 1979). Towards more "natural" interactive systems. *International Journal of Man-Machine Studies*, **11(3)**, pages 339-350.

D. Goodman (September 1987). *The Complete HyperCard Handbook*. Bantam Book.

S. Greenberg (1988). Tool use, reuse, and organization in command-driven interfaces. Research Report 88/336/48, 187 pages. PhD Dissertation, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada

S. Greenberg and I.H. Witten (May 15-19, 1988). How users repeat their actions on computers: Principles for design of history mechanisms. In *Proceedings of the CHI'88 Human Factors in Computer Systems*, pages 171-178, Washington, D.C. Also appears as Department of Computer Science, University of Calgary Technical Report No. 87/279/27 (February, 1987).

HCR Corporation (July 1987). Hi User's Guide (version 7.7). User Manual, 31 pages. HCR Corporation, 130 Bloor St. W., 10th Floor, Toronto, Ontario M5S 1N5.

P.E. Haeberli (June 1986). A data-flow manager for an interactive programming environment. In *Proceedings of 1986 Summer USENIX Technical Conference*.

D.C. Halbert (December 1984). Programming by Example. Technical Report No. OSD-T8402, 89 pages. Xerox Office Systems. PhD Dissertation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA.

S.J. Hanson, R.E. Kraut, and J.M. Farber (January 1984). Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems*, **2(1)**, pages 42-57.

P. Hayes and P.A. Szekely (September 1983). Graceful interaction through the COUSIN command interface. *International Journal of Man-Machine Studies*, **19(3)**, pages 19-30.

D.A. Henderson Jr. and S.K. Card (July 1986). The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics - Special Issue on User Interface Software Part II*, **5(3)**, pages 211-243.

W.N. Joy (November 1980). An Introduction to the C Shell. UNIX Programmer's Manual (7th edition) 2c: Supplementary Documentation (Virtual-VAX II Version), 46 pages.

W.N. Joy and D. Korn (July 1983). KSH - A shell programming language. In *Proceedings of USENIX Association Software Tools Users Group (Summer Conference)*, pages 191-202, Toronto, Ontario.

D. Kurlander and S. Feiner (October 10-12, 1988). Editable Graphical Histories. In *Proceedings of the 1988 IEEE Workshop on Visual Languages*, pages 127-134. IEEE Computer Society Press, Pittsburgh, Pennsylvania.

E. Lancaster-Thomas (June 1969). The Storing and Reuse of Real-Time Graphical Inputs. MSc Dissertation, 40 pages. Department of Electrical Engineering, Massachusetts Institute of Technology.

F.C.M. Lau and A. Asthana (March 1984). Yet another history mechanism for command interpretors. *SIGPLAN Notices*, **19(3)**, pages 51-56.

A. Lee (July 1987). User Support: Considerations, Features, and Issues. Technical Report No. CSRI-195: Office and Data Base Systems Research'87 F.H. Lochovsky (editors), pages 1-23. Computer Systems Research Institute, University of Toronto.

A. Lee (May 15-19, 1988). Study of command usage in three UNIX command interpreters. Interactive Poster. CHI'88, Washington, D.C.

A. Lee (August 21-25, 1989). Issues in design of history tool for user support. In *Proceedings of the Working Conference on User Interfaces (Engineering for Human-Computer Interaction)*, Napa Valley, CA.

A. Lee and F.H. Lochovsky (January 1990). User's command line reference behaviour: Locality versus recency. Technical Report No. CSRI-238: Office and Data Base Systems Research'89 F.H. Lochovsky (editors), pages 6-13. Computer Systems Research Institute, University of Toronto.

C. Linxi and A.N. Habermann (August 13, 1986). A history mechanism and undo/redo/reuse support in ALOE. Technical Research Report No. CMU-CS-86-148. Department of Computer Science, Carnegie Melon University.

D.L. Maulsby, I.H. Witten, and K.A. Kittlitz (August 1989). Metamouse: Specifying graphic procedures by example. *Computer Graphics*, **23(3)**, pages 127-136. Proceedings of the ACM SIGGRAPH'89, July 31 - August 4 1989, Boston, Massachusetts.

M. McMahon (1987). A practical system for managing mixed mode user interfaces. Working Paper, 29 pages. Symbolics, Inc., Cambridge, Massachusetts.

Y. Miyata and D.A. Norman (1986). Psychological issues in support of multiple activities. In D.A. Norman and S.W. Draper (editors), *User-Centered System Design: New Perspectives on Human-Computer Interaction*, pages 265-284. Lawrence Erlbaum Associates.

B.A. Myers (April 13-17, 1986). Visual programming, programming by example and program visualization: A taxonomy. In *Proceedings of the CHI'86 Human Factors in Computer Systems*, pages 59-66, Boston, Massachusetts.

B.A. Myers and W. Buxton (August 1986). Creating highly-interactive and graphical user interfaces by demonstration. *Computer Graphics*, **20(4)**, pages 249-258. Proceedings of the ACM SIGGRAPH'86, August 18-22 1986, Dallas, Texas.

J. Nievergelt and J. Weydert (1980). Sites, modes, and trails: Telling the user of an interactive system where he is, what he can do, and how to get places. In R.A. Guedj, P. ten Hagen, F.R. Hopgood, H. Tucker, and D.A. Duce (editors), *Methodology of Interaction*, pages 327-338. North-Holland.

D.R. Olsen and J.R. Dance (January 1988). Macros by Example in Graphical UIMS. *IEEE Computer Graphics and Applications*, **8(5)**, pages 68-78.

K.R. Paap and R.J. Roske-Hofstrand (1988). Design of Menus. In M. Helander (editors), *Handbook of Human-*

Computer Interaction, pages 205-235. Elsevier Science Publishers B.V.

R. Reichman-Adar (1986). Communication paradigms for a window system. In D.A. Norman and S.W. Draper (editors), *User-Centered System Design: New Perspectives on Human-Computer Interaction*, pages 285-313. Lawrence Erlbaum Associates.

E. Rich (March 1983). Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, **18(3)**, pages 199-214.

SUN Microsystems, Inc (February 17, 1986). *Windows and window based tools: Beginner's guide*. SUN Microsystems, Inc., 2550 Garcia Avenue, Mountain View, CA 94043.

J. Scofield (November 1981). Editing as a Paradigm for User Interaction A Thesis Proposal. Technical Report No. 81-11-01, 27 pages. Department of Computer Science, University of Washington.

H. Senay (August 21-25, 1989). Fuzzy command grammars for user modelling in intelligent interfaces. In *Proceedings of the Working Conference on User Interfaces (Engineering for Human-Computer Interaction)*, Napa Valley, CA.

R. M. Stallman (Spring/Summer 1981). EMACS - The extensible, customizable self-documenting display editor. *SIGOA Newsletter*, **2(1-2)**, pages 147-156. Proceedings of the ACM SIGPLAN/SIGOA Symposium on Text Manipulation, June 8-10 1981, Portland, Oregon.

G.R. Stearns (August 1989). Agents and the HP NewWave application program interface. *Hewlett-Packard Journal*, pages 32-37.

J.L. Steffen and M.T. Veach (July 1983). The edit shell - Combining screen editing and the history List. In *Proceedings of USENIX Association Software Tools Users Group (Summer Conference)*, pages 187-190, Toronto, Ontario.

W. Teitelman (August 22-25, 1977). A display oriented programmer's assistant. In *Proceedings of IJCAI-77, 5th International Joint Conference on Artificial Intelligence*, Volume 2 , pages 905-915. A longer version appears in Xerox PARC Technical Report No. CSL-77-3 (November 1982).

W. Teitelman (April 1984). A tour through cedar. *IEEE Software*. Also appears in Xerox PARC Technical Report No. CSL-83-11 (June 1984), pp. 23-88.

W. Teitelman and L. Masinter (April 1981). The interlisp programming environment. *IEEE Computer*, pages 39-50.

S.W. Tyler and S. Treu (October 6-8, 1986). Adaptive interface design: A symmetric model and a knowledge-based implementation. *SIGOIS Bulletin*, **7(2-3)**, pages 53-60. Proceedings of the Office Information Systems.

J.S. Vitter (October 1984). US&R: A new framework for redoing. *IEEE Software*, **1(4)**, pages 39-52.

M. Whitby (July 1986). See MAC run. *MacUser*, **1(10)**, pages 38-42.

I.H. Witten, J.G. Cleary, and J.J. Darragh (April 1983). The Reactive Keyboard: A new technology for text entry. Technical Research Report No. 83/121/10, 7 pages. Department of Computer Science, University of Calgary.

M. Young, R.N. Taylor, and D.B. Troup (June 1988). Software environment architectures and user interface facilities. *IEEE Transactions on Software Engineering*, **14(6)**, pages 697-708.

P.T. Zellweger (April20-22, 1988). Active Paths Through Multimedia Documents. In J.C. van Vliet (editors), *Document Manipulation and Typography*, pages 19-34. Cambridge University Press.