Chapter 2

Uses of Information in a History

This chapter presents seven basic uses of the information in a user's history which can potentially enhance user-computer interactions. These uses are based on a survey of a number of history-like tools available in current systems. A taxonomy is used to structure the presentation of these uses, their various manifestations, and their system support. In addition, the state of current history tools is examined with respect to 1) user needs for each manifestation of history use and on system capabilities provided in support of each and 2) support of these uses of history in a number of general-purpose development systems.

Before proceeding, we need to clarify our use of several terms in this dissertation. The terms *history* and *user history* are used interchangeably to refer to a collection of information recorded from an earlier part of a user's interactions. The term *script* refers to a sequence of user actions to be carried out [Archer, Conway, & Schneider, 1984]. If in the course of script recording, user actions are actually performed, then the recording (aside from being a script) is part of a history. A future instantiation of a script is a *future part of a history*. Viewing a user's interactions with respect to an activity timeline, actions and objects that took part earlier are candidates for a history and actions and objects that are to be performed in the future are candidates for a future part of a history.

Taxonomy

There have been two previous surveys of history tools. Greenberg (1988b) examined different interaction styles in history tools for facilitating the reuse of computer tools. Linxi and Habermann (1986) examined different uses of the information in a history to facilitate the software development process. Like Greenberg (1988b), we are interested in how history tools facilitate user interactions but like Linxi and Habermann (1986), our survey enumerates different uses of history information. Our survey results include uses of history information that are not evident in the software development task domain. Furthermore, we partition our survey results into a taxonomy of *uses of history*. Each *use of history* is an abstraction of a designer's conception of a user's *usage intention* (e.g., reuse or error recovery).

Our taxonomy characterizes each *use of history* in terms of two variables: *type* and *system support* . Examples of systems are provided for the various system support associated with a particular type of history use (see Table 1).

Uses	Type	System Support	Example Systems
reuse	a script literally		CONMAN [Haeberli, 1986] PLAYER PIANO [Bier & Freedman, 1985]
	a previous interaction with possible modifications	list of accepted operations descriptive manipulation	C Shell [Joy, 1980], INTERLISP -D USE , REDO [Teitelman & Masinter, 1981]
		direct manipulation	KORN Shell [Korn, 1983], HCR HI [HCR Corporation, 1987], MINIT [Barnes & Bovey, 1986], TC Shell [Ellis, Greer, Placeway, & Zachariassen, 1987], INTERLISP -D FIX [Teitelman & Masinter, 1981]
		display's contents	CEDAR [Teitelman, 1985], EMACS [Stallman, 1981], SMALLTALK -80 [Goldberg, 1984], SUN TOOLS [SUN Microsystems, Inc., 1986]
	operations as a functional group	macro history macro recorded macro	ALOE [Linxi & Habermann, 1986] EMACS [Stallman, 1981], HP NEW WAVE AGENT [Stearns, 1989], MACROS BY EXAMPLE [Olsen & Dance, 1988], MEXEC [Ash, 1981a], TEMPO [Whitby, 1986], QUIC KEYS [Bobker, 1988]
		programming with example by example	SMALL STAR [Halbert, 1984] METAMOUSE [Maulsby, Witten, & Kittlitz, 1989] PERIDOT [Myers & Buxton, 1986]
inter-referential I/O	relate previous I/O to current input		SYMBOLICS [McMahon, 1987]
error recovery	system recovery	restore system state undo erroneous operations	SMALLTALK -80 [Goldberg, 1984] US&R [Vitter, 1984], EMACS [Stallman, 1981], CHIMERA [Kurlander & Feiner, 1988]
	command correction	descriptive manipulation direct manipulation	C Shell [Joy, 1980] INTERLISP -D HISTMENU & FIX [Teitelman & Masinter, 1981], KORN Shell [Korn, 1983], TC Shell [Ellis, Greer, Placeway, & Zachariassen, 1987]
navigation	where am I, where did I come from, where have I been	information spaces	HYPER CARD RECENT [Goodman, 1987], TIMELINE PAGE [Feiner, Nagy, & van Dam, 1982], WHAT , WHERE , WHENCE [Engel, Andriessen, & Schmitz, 1983]
		activity spaces	SITES , MODES , and TRAILS [Nievergelt & Weydert, 1980], Room Stack in ROOM MODEL [Chan, 1984], Back Door in ROOMS [Card & Henderson, 1987]
reminding	user consultation		SUN TOOLS [SUN Microsystems, Inc., 1986], JOBS [Joy, 1980], SITES , MODES , and TRAILS [Nievergelt & Weydert, 1980]
	visual cues	display user history	INTERLISP -D HISTMENU [Teitelman & Masinter, 1981], SUN TOOLS [SUN Microsystems, Inc., 1986], HYPER CARD RECENT [Goodman, 1987], TIMELINE PAGE [Feiner, Nagy, & van Dam, 1982]
user modelling			UKNOW [Desmarais & Pavel, 1987], UNIX CONSULTANT [Chin, 1986], STEREOTYPE [Rich, 1983]
user interface adaptation		default menu selection predict next action	SUN TOOLS [SUN Microsystems, Inc., 1986] REACTIVE KEYBOARD [Witten, Cleary, & Darragh, 1983]

Table 1: Taxonomy of uses of a history along with example systems.

The *type* variable characterizes different manifestations of a usage intention. To illustrate, reuse may be manifested as the literal reuse of a group of user actions – *reuse a script literally* – or as the reuse of an object/action with some modifications – *reuse a previous interaction with possible modifications* . Each manifestation suggests a set of user needs. For example, users who want to *reuse operations as a functional group* need to group several operations together for invocation as a unit.

The *system support* variable characterizes system capabilities that are provided to meet certain user needs of a particular type of history use. System support varies in terms of the sophistication of the user support provided and reflects the amount of user effort required to realize the particular type of use of history. For example, some designers provide a *macro* facility to support *reuse operations as a functional group* , while other designers provide an *example programming* facility. A macro allows users to associate a sequence of actions to a single action while an example programming facility allows users to generalize an example execution trace to a program.

Our taxonomy is the result of a survey of current history-like tools that directly enhance a user's interactions with a system. While one may envision different types of uses of history to support user interactions, the taxonomy only considers those supported by existing systems and those uses of history that fall inside the scope of interest. For example, user traces for analyzing the design of a system are not considered a direct user support capability.

The taxonomy characterizes the value of the information maintained in a history in terms of seven basic usage intentions. The taxonomy provides a framework for presenting and critiquing user needs and system support for a particular type of use of history. Furthermore, the usage intentions and their various manifestations and implementations provide a context for examining how well current general-purpose systems support these seven uses of history. Finally, the taxonomy provides a structure for thinking about the kinds of information and functions required to use such history information. The next chapter presents the insights into design requirements for multi-use history tools extracted from the taxonomy.

History for Reuse

Currently, the most common use of a history tool is to reuse and possibly modify a history item to save keystrokes and/or mouse strokes [Linxi & Habermann, 1986]. Different types of reuse are supported in current history tools.

Reuse a Script Literally

An early form of reuse originates from early scripting systems where the same sequence of actions – recorded during the performance of an earlier task – is replayed literally in the current context. The recording ''pushes the buttons'' as if the user is there doing it – similar to a player piano. Since it is intended for situations in which the same operations are to be performed each and every time, this type of reuse is both primitive and limited.

In a *programming by example* system, the system makes the procedural inferences and generalizations based on observations of user actions for a number of related example tasks and knowledge about the specific domain (e.g., METAMOUSE deduces graphical procedures from user execution traces in a drawing program [Maulsby, Witten, & Kittlitz, 1989]).

History for Inter-referential I/O

In human dialogue, conversants make abbreviated references to objects and actions that took place earlier in the dialogue [Draper, 1986; Reichman-Adar, 1986]. For example, two conversants have just attended a talk given by speaker A and in their subsequent conversation about the talk, they use the referent, *the talk* , to refer to the talk instead of using an elaborated reference. Similarly, two persons (A and B) are examining the contents of a file directory on a computer system and person A decides to relocate one of the files to another file directory and then person B tells person A to *do the same* for another file; a request which person A can interpret.

Inter-referential I/O is the computing analogue of this conversational capability [Draper, 1986]. It allows the user's input or the system's output to reference previous input and/or output available on the display. Current systems support part of this capability. Specifically, users can make explicit the links between actions and objects from an earlier part of the dialogue with those from the current part of the dialogue (i.e., *relate previous input/output to current input*). This enables users to make abbreviated associations which the system can disambiguate and interpret. However, users are unable to relate previous input/output to current output.

This use of history involves more than the simple reuse of parts of a user history, it allows a user to use abbreviated referents to relate pieces of past and present interactions. Such tools are able to track and interpret these referents because they share a common interaction context with the user much like the shared context between two human conversants. Hence, user references can be explicitly abbreviated and systems are able to disambiguate them.

The SYMBOLICS GENERA programming environment [McMahon, 1987] highlights display objects that are suitable as inputs to the current user command as the user moves the mouse over them (see Figure 5). This is made possible by a type mechanism which relates application objects, including those on the display, with the way they are to be used in particular user interface situations. Thus, the system has a structured context and an abbreviated referencing scheme for linking objects required in a user action with those on the display much like the shared context and abbreviated referents available in human conversations.

History for Error Recovery

Error recovery tools allow users to recover from errors (e.g., typing). Two different types of error recovery are intended and supported: *command correction* and *system recovery*. Command correction is simpler than system recovery.

- Remove erroneous commands
- Insert missing commands
- Substitute for erroneous commands
- Re-arrange the order of commands
- Change of heart (UNDO an UNDO)

A history contains the primitive user commands that have been issued and is maintained as a tree-like data structure with a pointer variable. The commands, UNDO, REDO, and SKIP, allow users to undo, redo, and skip over previously issued commands by reorganizing the links in the tree-like history data structure. Figure 6 illustrates two sets of error recovery scenarios using these commands.

History for Navigation

In the course of using a computer system, users commonly ask questions like where they are, where they just came from, and where they have been [Engel, Andriessen, & Schmitz, 1983; Fitter, 1979; Nievergelt & Weydert, 1980]. The information in a history helps users to answer these questions. Depending on whether the user is navigating through an *information space* or *activity space*, a system provides different navigation aids.

In navigating through large information spaces, users can become lost [Mantei, 1982]. Thus, such systems typically maintain system snapshots corresponding to places a user visited, generally in chronological order (see Figure 7).

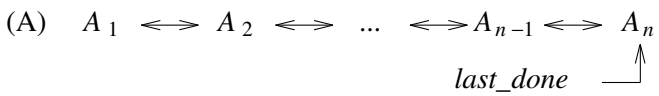
An activity space contains all user activities for a particular task. Users are known to juggle several tasks by moving back and forth between activity spaces. Occasionally, they lose track of their task switching [Bannon, Cypher, Greenspan, & Monty, 1983; Card & Henderson, 1987; Cypher, 1986; Miyata & Norman, 1986]. ROOMS' BACK DOOR [Card & Henderson, 1987] allows users to find out about the last activity space they worked in while ROOM MODEL's ROOM STACK [Chan, 1984] lists the activity spaces the user visited in chronological order.

History for Rem inding

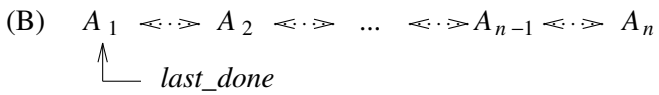
The contents of a history may remind users of past knowledge and events. A reminding is deliberate or spontaneous activation of past knowledge and experiences and integration of that information into the current context [Ross, 1989]. A deliberate reminding is supported by *user consultation* of a history while a spontaneous reminding is supported by *visual cues* in a history. The effectiveness with which history tools support these two types of reminding depend largely on whether the tools provide good visual cues as well as browsing and querying tools.

User Consultation

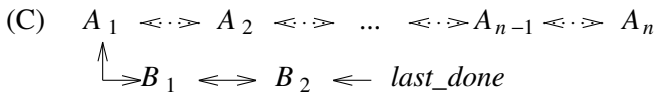
A deliberate reminding occurs when users query or consult their history to find specific information. For example, users can examine their history:



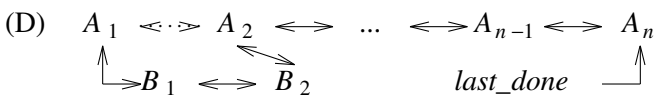
The user realizes after issuing commands $A_1, A_2, \ldots, A_n$ that commands B_1 and B_2 must be inserted between A_1 and A_2.



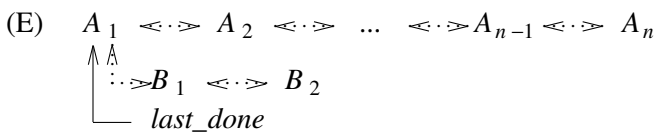
Commands $A_n, A_{n-1}, \ldots, A_2$ are undone (signified by dotted arrows).



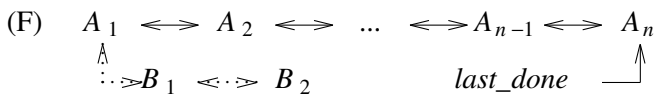
Commands B_1 and B_2 are issued. They appear as new nodes on a new branch emanating from the node A_1 with solid arrows.



Commands $A_2, \ldots, A_n$ are reinstated with n-1 REDOs. At this point, the user realizes that the original sequence in step (A) is the correct sequence.



The user undoes commands following A_1.



The user invokes REDO to redo A_2 and then issues n-2 REDOs to redo commands following A_2.

Figure 6: (A) illustrates the state of the tree-like history data structure before the user performs a sequence of changes (B) to (F) related to two error recovery operations. (B) to (D) illustrates insertions of two missing commands. (E) to (F) illustrates a change of heart; the user wants to undo changes made in steps (B) to (D). Note, commands with dotted arrows are undone while commands with solid arrows are issued.

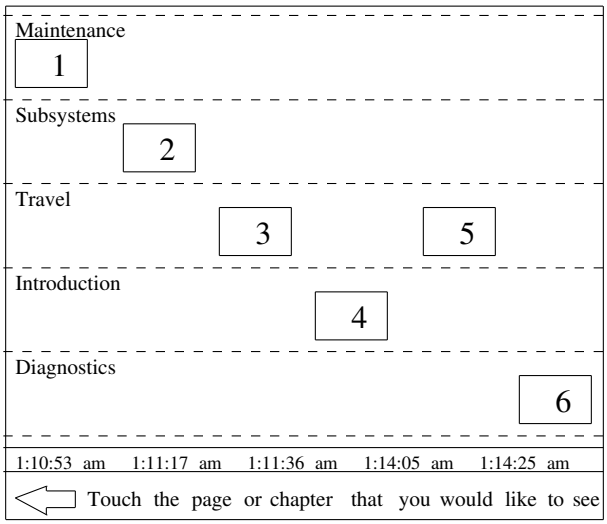


Figure 7: In Feiner, Nagy, and van Dam (1982)'s system, the last 6 examined pages are displayed chronologically as miniatures along their parent chapter's band. Selecting the miniature returns the user to that page. Other pages in the timeline are displayed by selecting the scroll arrow.

- to identify activities requiring attention or causing errors,
- to find relevant information from a previously related task, or
- to re-acquire the mental context for an interrupted activity.

An important design consideration is the user effort required to cull the information from a history and the extent of the system support available to assist such user efforts (e.g., query and browse). Without appropriate system support, this type of history for reminding is limited and users must expend considerable effort to extract the desired information.

Visual Cues

The display of a user history provides visual cues that trigger the spontaneous recall of relevant past information and activities that the user would otherwise not remember. More importantly, reminders of past activities may lead a user to consult the history in more detail; an activity which would never have been pursued without the benefit of the reminding. While a history provides the information, visual cues are crucial to the spontaneous activation of a reminding. Thus, it is important to provide appropriate support for such visual cues and to develop innovative and effective visual representation schemes. Current history-tool support for this history use consists largely of displaying portions of the user history with little or no novel visual cues.

History for User Modelling

A user history may contain information about a user's skill level, task knowledge, preferences, and personality traits. When systems infer such information from a history to formulate user models, they are using *history for user modelling* [Chin, 1986; Desmarais & Pavel, 1987; Rich, 1983; Tyler & Treu, 1986]. The user model may then be used to adapt system behaviour or to provide user-tailored assistance. Figure 8 illustrates the kinds of information that UKNOW infers from a user history [Desmarais & Pavel, 1987]. Systems use various techniques to extract user information including deterministic and probabilistic approaches, behaviour to structure transformations, and induction and knowledge inferences. Current history tools for user modelling are limited by our understanding of user characteristics that differentiate individuals and by the techniques for inferring user characteristics.

History for User Interface Adaptation

Using a user history and appropriate heuristics, a system can adapt system behaviour to suit user needs, to predict user actions, or to infer user preferences [Greenberg & Witten, 1988b]. A simple example is a menu system whose default selection is the last menu selection. Another example is the REACTIVE KEYBOARD which predicts what the user is going to type based on an adaptive model of the text that has been typed previously [Witten, Cleary, & Darragh, 1983].

Like programming-by-example systems, user-interface-adaptation tools are successful within specific application environments because they can exploit application knowledge and use knowledge-based techniques to interpret user behaviours. However, it is difficult to exploit application knowledge in a general-purpose environment. Therefore, general-purpose history tools for user interface adaptation have not fared as well. A more viable approach is to use knowledge-based or probabilistic techniques to infer behavioural semantics from a user history (e.g., use rules to interpret observed behaviour [Quinn & Russell, 1986]).

Assessm ent

While the preceding section presented user needs and critiqued the level of system support available for different types of uses of history, this section examines the uses of history that are directly supported by current systems. To facilitate this examination, a number of interactive systems are selected and assessed in terms of how well they directly support all seven uses of history.

In order to provide a fair assessment of the state of current support for all uses of history, only general-purpose development environments are considered rather than special-purpose systems (e.g., text editors, information retrieval systems). Such environments support access to different applications (e.g., text editing, graphical drawing, electronic communication, programming, and managing file systems). We picked six systems that supported some form of a history tool for our assessment: ALOE, INTERLISP-D , MINIT , MPW , ROOMS , and SEED. Appendix A provides a brief description of these systems.

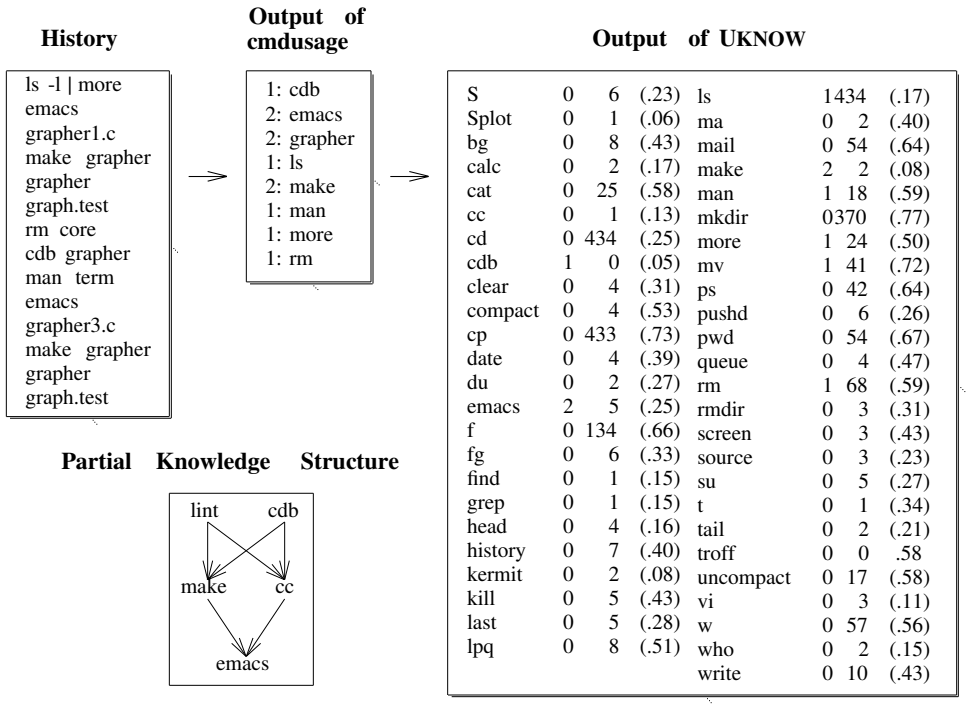


Figure 8: UKNOW uses three techniques to deduce commands known by a user: observation, inference, conditional probability [Desmarais & Pavel, 1987]. The first two techniques are based on analyses of a user history. One metric, appearing in the first column of numbers in the UKNOW output and in the ‘‘cmdusage’’ output, is the observed frequency of use of each command. A second metric, appearing in the second column of numbers in the UKNOW output and derived with the aid of a knowledge structure, is the number of inferences about a user’s mastery of a given command based on a user’s mastery of other commands. A knowledge structure links sets of UNIX commands together with implication relationships. In the partial knowledge structure, observation of *lint* will result in one inference of mastery to both *make* and *cc* and in two inferences to *emacs* . The third metric is the conditional probability that a command is known by users given that their mastery of the command is neither observed nor inferred (note that the first two metrics are 0). Mastery of *troff* was determined using this conditional-probability measurement technique (i.e., third metric is not enclosed by parentheses to indicate that its value is derived from the conditional probability).

Results and Discussion

Each system is assessed in terms of whether it *does* or *does not* directly support each type of use of history. Table 2 summarizes the results of the assessment. At a glance, no single system supports all seven uses of history. These systems support some manifestation of reuse, error recovery, navigation and reminding.

Uses of History	ALOE	INTERLISP-D	MINIT	MPW	ROOMS	SEED
reuse						
operations literally	●	○	○	○	○	●
a previous interaction with possible modifications	●	●	●	●	●	●
operations as a functional group	●	○	○	○	○	○
inter-referential I/O	○	○	○	○	○	○
error recovery						
command correction	●	●	●	●	●	●
system recovery	●	●	○	○	●	●
navigation						
where am I	○	○	○	●	●	○
where did I come from	○	○	○	○	●	○
where have I been	○	○	○	○	○	○
user remindings						
user consultations	●	●	●	●	●	●
visual cues	○	○	○	○	○	○
user modelling	○	○	○	○	○	○
adaptive user interfacing	○	○	○	○	○	○

Table 2: Assessment of the uses of history information directly supported by eight general-purpose environments. A ● means the *system does support* the use while ○ means the *system does not support* the use.

History for user modelling, user interface adaptation, and inter-referential I/O are not supported by any of the six systems. In order to support such uses of history, a system must maintain application-specific information. Typically, such information is not maintained by a general-purpose development environment.

History tools also need to support the various types of history uses; otherwise, users may have to do a lot of fiddling to mimic these uses. For example, users who want to *reuse operations as a functional unit* but are provided with a tool to *reuse a previous interaction with possible modifications* must deal with each operation in the set individually rather than the set as a whole. All six systems support reuse of previous interactions with possible modifications, command correction, and deliberate reminding. Reminding is facilitated by history tools when these tools allow users to consult their history visually. ALOE nd SEED are the only systems to support more than one type of history for reuse while MPW and ROOMS are the only systems to support some type of history for navigation. All systems, except MINIT and MPW, support both types of history for error recovery.

Current general-purpose history tools fail to support all seven basic history uses as well as all types of history uses within the seven basic categories. This failure is due to two factors: the kind of information maintained in a history and the history capabilities provided. An in-depth discussion of both factors is deferred until the next chapter.

Concluding Rem arks

This chapter presents three important observations about the current state of history-like tools revealed by a survey of such tools. First, there are seven different basic reasons for using a history:

(1) *History for reuse* – Users repeat operations to avoid re-entering operands and operations from scratch.

(2) *History for inter-referential I/O* – Users make abbreviated references to objects and actions that took place in an earlier part of the dialogue.

(3) *History for error recovery* – Users edit an incorrect action or undo the effects of an erroneous operation on a system.

(4) *History for navigation* – Users consult a history to find out where they have been, where they have just come from, and where they are.

(5) *History for reminding* – Users are reminded of relevant information associated with past activities.

(6) *History for user modelling* – Systems infer or derive information about users by observing their activities in the course of their interactions.

(7) *History for user interface adaptation* – Systems use history as a source of information to adapt automatically the interface behaviour to the user's needs.

Second, the taxonomy variables, *type* and *system support* , are used to characterize a use of history. The *type* variable characterizes the various manifestations of a basic usage intention. The *system support* variable characterizes capabilities that system designers have provided to address particular user needs. Both variables are useful, respectively, for thinking about the user needs posed by a particular type of history use and for thinking about the degree of system sophistication that is needed to support a particular type of history use.

Third, the taxonomy also provides a global context for assessing the extent to which current systems support the seven uses of history listed in Table 1. The assessment of six general-purpose development systems reveals that current general-purpose history tools fail to support all seven uses of history. Thus, users must either perform the desired operations manually or expend considerable effort to make the desired use of history fit the supported use of history.

This chapter introduces concepts that are relevant to the discussion in the next chapter concerning issues related to the design of history tools. Thus, the taxonomy provides a vehicle for not only critiquing current history tool designs but also thinking about the kinds of information a history tool should maintain and the capabilities a history tool should support.