Chapter 1

Introduction

The field of *user support* is concerned with how to support user-computer interactions. Past efforts to address this problem have taken one of three approaches: user-interface design, run-time support, and user instruction. User-interface design stresses the importance of designing usable interfaces because usability is an important user support concern. Run-time support focuses on the provision of run-time tools and resources, computer-based and non-computer-based, because user interactions are complex and difficult and user-interface issues are not always resolved at design time. Through run-time support, designers carry over their concern for users from user-interface design to the use of computer systems. A third approach involves developing methodologies for instructing and training users on the use of computer systems. The premise is that user-computer interactions are enhanced with user instructions.

In this thesis, we are concerned mainly with the provision of run-time support. Past efforts have concentrated on three aspects of user interactions: *user-centered*, *task-centered*, and *interaction-centered* views. The user-centered view is concerned with issues related to user intentions and actions and in particular, cognitive, behavioural, and social issues associated with user-computer interactions. The task-centered view is concerned with issues related to task analysis, task description, and task-to-tool mappings [Croft, 1984]. The interaction-centered view is concerned with issues related to needs and protocols for effective communication [Hayes, Ball, & Reddy, 1981; Nickerson, 1976; Taylor, 1988a, 1988b; Thimbleby, 1980]. While the distinction between interaction-centered and user-centered views and between interaction-centered and task-centered views may not be clear cut, the distinction between user-centered and task-centered views is clear. In the task-centered view, the task drives the design of an interface while in the user-centered view, user needs drive the design of an interface and interface needs drive the design of the rest of a system [Norman, 1986].

Since we cannot hope to address all of the various user support concerns, our discussion focuses on user-centered, computer-based, run-time user support. Furthermore, our treatment of this aspect of user support is biased towards cognitive and behavioural issues related to the provision of tools and resources for smoothly integrating computing with user tasks. However, this is not to suggest that other issues like implementation, social and organizational context of computing, theory of task domains, and training of users are unimportant.

This chapter presents our motivations for focusing on user-centered, computer-based, run-time user support. Then, we examine three important considerations for user support. They highlight the multiplicity of concerns and challenges that underlie attempts to provide user support. Finally, we propose that history, and tools based on history, can address many of these user support concerns and challenges. Several indications of the value of history tools are presented followed by a problem statement and brief overview of the dissertation.

Motivation

Computer users can encounter difficulties in interacting with a computer system and therefore the system must enable users to deal with and manage these difficulties as they arise [Brown & Newman, 1985; Quinn & Russell, 1986]. There are three reasons why a user-centered, computer-based, run-time support approach is important for dealing with such difficulties: diverse user knowledge and experience, task complexity, and interaction barriers.

First, support needs to be *user-centered* because users have diverse computer knowledge and experience. Thus, system designers need to provide user-centered tools and resources that will address individual user needs and concerns. Second, support needs to be *computer-based* because users perform complex computing tasks. Thus, system designers need to provide computer-based tools and resources that will allow them to focus on conceptual and difficult aspects of performing such tasks and to accomplish them easily [Card, 1989; Gasser, 1986; Rissland, 1984]. Finally, support needs to be *run-time* because the powerful and sophisticated tools in computer systems present formidable barriers to a user's understanding and use of these systems [Hayes, Ball, & Reddy, 1981; Winograd, 1973]. Thus, system designers need to provide run-time tools and resources to help users overcome the ''communication'' and ''utility'' barriers as they arise.

Considerations in the Design of User Support

In order to provide effective run-time assistance to users, designers need to be aware of and to understand users, their behaviours, and their interactions with computers. A literature review reveals three important considerations:

- Users are individuals.
- User behaviours are variable.
- User-computer interactions are complex and difficult.

Users are Individuals

Studies of user task performance and system usage consistently show that user behaviour in user-computer interactions is non-homogeneous [Boies, 1974; Card, Moran, & Newell, 1983; Carey, 1982; Egan, 1988; Greenberg & Witten, 1988a; Hanson, Kraut, & Farber, 1984; Potosnak, Hayes, Rosson, Schneider, & Whiteside, 1986]. User classification schemes have been proposed to illustrate

and to characterize individual differences and traits [Carey, 1982; Norman, 1984; Potosnak, Hayes, Rosson, Schneider, & Whiteside, 1986]. Each class represents an interpretation of user behaviour and a characterization of distinctive user differences (e.g., cognitive factors, preferences, frequency of system use). These classifications are not absolute as certain implicit generalizations and assumptions about individuals are made.

User Behaviours are Variable

Studies reveal that user behaviours vary from situation to situation. Such behavioural variability is supported by research into learning, human cognition, interaction gulfs, and engagement.

Learning research reveals that, with training and exposure to a task, a user's skill evolves from a *problem-solving* stage to a *procedural* stage and then to an *automatic* stage [Anderson, 1982; Card, Moran, & Newell, 1983; Rasmussen, 1983]. In unfamiliar stages of task performance, users with minimal task knowledge rely on heuristic strategies to search and navigate a problem space (i.e., problem solve). As user skills develop, operator sequences for moving from problem state to problem state are more predictable and are integrated and composed into debugged cognitive structures, known as *mental schema* . Mental schema knowledge allows a problem solver to recognize a problem state and its associated operators. It is encoded as procedural rules and then, with further refinement, human behaviour can be described as if these rules are collapsed into methods controlled by a single rule. As tasks become well-practiced, actions become automatic, and users are unable to describe the rules for controlling their task performance. Depending on which stage users are at, their behaviour varies from one task exposure to another.

Human information processing resources and capacities constrain human cognition [Card, Moran, & Newell, 1986; Lindsay & Norman, 1977]. Such constraints encourage an intuitive preference for mental processing strategies that lead to sequences of simple operations [Rasmussen, 1983]. Strategies may involve fitting task activities to system functions, optimizing for transfer of previous task results, and minimizing the need for new information. These constraints, coupled with different human information processing demands imposed by different situations, lead to variability in observed user behaviours.

Gulfs between a person's goals and knowledge and a system's requirements are qualified in terms of two distances: *semantic* and *articulatory* distances [Hutchins, Hollan, & Norman, 1986; Norman, 1986]. *Semantic distance* is the distance between how users think of their tasks and the semantics of the interface language. *Articulatory distance* is the distance between the semantics of the interface language and the physical form of the interface language. The less natural the relationship between the two ends of a distance, the less natural the formulation or articulation of a user intention or user action and the less natural the interpretation or evaluation of system semantics or system response. Hence, users have more problems and their behaviour is more variable.

Finally, *engagement* is the qualitative feeling of manipulating interface objects [Hutchins, Hollan, & Norman, 1986]. Variability in user behaviour arises depending on how users engage a system. There are situations where user actions

and objects may be described precisely with an abstraction (i.e., *indirect engagement*) rather than direct manipulation (i.e., *direct engagement*) or vice versa and as a result, differences in user behaviour are observed. Current interfaces do not effectively integrate both direct and indirect engagement.

User-Computer Interactions are Complex and Difficult

User-computer interactions do not always proceed in a straightforward manner. User interface and task complexities, individual differences, and behavioural variability often make on-going user interactions complex and difficult. Behavioural studies have shown that user interactions include four kinds of activities: multi-tasking, improvisation, repetition, and error correction.

Studies have found that user activities do not always proceed in a serial and uninterrupted fashion. Users often engage in a number of activities by either processing them concurrently or digressing and switching back to an activity [Bannon, Cypher, Greenspan, & Monty, 1983; Cypher, 1986; Greenberg & Witten, 1985; Henderson & Card, 1986a; Stephenson, 1973]. In switching amongst several parallel activities, users internalize complex sequences of parallel activities to run tasks sequentially on a computer. Factors such as limited resources and structural and functional mismatches between human and computer systems, extent of control of mental processing, nature of tasks (e.g., problematic, routine, well-defined), and human characteristics affect a user's ability to perform multiple activities [Cypher, 1986; Gasser, 1986; Lee, 1992a; Lindsay & Norman, 1977; Miyata & Norman, 1986; Norman & Shallice, 1986; Suchman, 1983].

According to Suchman (1987), user actions are *situated* : all activities take place in some specific on-going concrete situation that demands immediate attention. Real situations are typically rife with uncertainties, complexities, and contingencies. They arise for any number of reasons including misalignment between computing and work processes, work contingencies, system design restrictions, poor functionality, skill level, and nature of tasks [Gasser, 1986]. Users must continually evaluate and decide what to do next, utilizing resources from the immediate surroundings; they need to improvise [Agre & Chapman, 1990]. Thus, user actions may be systematic but they are not planned in the traditional sense.

Studies of computing and non-computing behaviours observed repetition in user activities. Greenberg and Witten (1988b), Hanson, Kraut, and Farber (1984) and Krishnamurthy (1987) reported that a few commands in a user session constitute a large percentage of the commands issued. Greenberg (1988b) also noted repetitive behaviours in the form of replaying favourite songs, looking up information in manuals several times, favouring certain physical tools or cooking recipes. Recurrent activities arise for several reasons: problematic tasks necessitate a trial and error process; tasks are routine and frequent (e.g., users invoke a window system every time they log in) or inherently repetitive (e.g., deleting files in different directories); and a number of tasks are performed using the same actions because less effort is required or poorly designed systems force users to do so.

Finally, users are fallible even in a well-designed computer system. Human error falls into two categories: *mistakes* and *slips* [Norman, 1981; Norman, 1983]. *Mistakes* are errors resulting from the formulation of wrong intentions while *slips* are errors resulting from the execution of inappropriate actions for an intention.

Mistakes include errors associated with faulty decision making and misunderstandings, misinterpretations, or misdiagnosis of situations while using a system [Norman, 1983]. Examples include mistaking a system state (i.e., mode errors) or specifying an incomplete or ambiguous intention (i.e., description errors). Slips are attributed to unintentional activation of actions, false activation of actions, or failure to trigger correctly selected actions.

Implications for User Support Endeavours

The preceding observations about users, their behaviour, and their interactions with computers highlight several important concerns for the provision of user support. No single mechanism is available to address all the concerns but there are three basic user support capabilities that are desirable and are discussed briefly in this section: *user models*, *adaptation*, and *facilitation* [Lee, 1992a].

A *user model* contains such information as a user's preferences, knowledge, characteristics, intentions, behaviours, and performance capabilities [Benyon, 1984; Chin, 1986; Quinn & Russell, 1986; Rich, 1983; Self, 1974; Sleeman, 1985; Witten, Greenberg, & Cleary, 1983]. The user information allows the interface to be aware of individual user needs and requirements. As a result, a system can better accommodate individual differences. *Adaptation* allows interfaces to adapt system behaviours to changing user needs and to reduce the amount of user effort that is expended. Therefore, a system can better accommodate variable user behaviours. *Facilitation* eases the cognitive and physical overhead associated with user interactions and ensures that user interactions proceed relatively smoothly. It is an important component of computer systems because it addresses concerns related to the support of complex and difficult user interactions.

History Tools : A Proposed User-Support Candidate

Computer-based user support is available, generally, in the form of help systems [Elkerton, 1988; Lee, 1992a]. This thesis proposes a different user support facility known as a *history tool*. A *history* (also known as user history) is a log of a user's past interactions. A *history tool* permits users to refer to their history and to incorporate parts of their history into the context of their current interactions. There are several reasons why history-based, user support tools are important and merit investigation:

- User support capabilities rely on history information.
- Humans draw heavily from past experiences.
- History-cued problem solving facilitates learning.
- History information helps users cope with improvisation.
- History tools relieve cognitive and physical burdens.

User Support Capabilities Rely on History Information

A history is a valuable source of information for the three basic user support capabilities (i.e., user models, adaptation, and facilitation). It contains descriptive, interpretive, and functional information. Descriptive information is fundamental to the operation of user models. It includes, for example, preferences, style of interaction, user skill level, and task knowledge. Interpretive information helps users to diagnose, explain, and understand problematic situations. It includes, for example, errors, feedback, and diagnostics. Functional information provides users with the means to adapt system behaviour and to facilitate user-computer interactions. It includes, for example, information about how users interact with a system, how they adapt to a system, and how a system facilitates the execution of user tasks. The interpretive and functional types of information are useful for adaptation and facilitation.

Systems have used history information as a crucial element of user support tools. However, these efforts have focused on identifying the functionality of history tools rather than characterizing the types of history information and the uses of history information. We argue that such characterizations are important as they reveal the value of and the range of uses of history information.

Humans Draw Heavily from Past Experiences

Humans draw heavily from their vast store of knowledge and experiences. This human characteristic is exploited by two different machine learning research efforts. In *case-based reasoning* , a reasoner uses memory of past cases within the same domain, sharing similarities to the current situation, to interpret or to solve a new case [Rissland, Kolodner, & Waltz, 1989]. *Reasoning by analogy* , like case-based reasoning, uses analogous situations, but not necessarily from the same domain, as the basis for reasoning about a problem.

In the user support context, a history contains information about past interactions. This information can be used to improve the quality and efficiency of user interactions. Users can avoid past errors, forego fruitless approaches attempted previously, or bootstrap onto a previous solution and thereby focus quickly on important aspects of a solution which requires user attention.

History-Cued Problem Solving Facilitates Learning

Studies of problem-solving mechanisms reveal that the cognitive processes required for problem solving and learning do not overlap sufficiently for learning to occur during problem-solving [Sweller, 1988]. In fact, Sweller (1988) suggests that a *means-ends* problem-solving strategy interferes with learning by preventing problem solvers from learning essential aspects of a problem's structure. A means-ends strategy involves working backward iteratively from a goal state by setting subgoal states until the problem state is reached. It is extremely effective for problems in which problem states can be specified as goals because it encourages a problem-solver to attend to differences between a current problem state and a goal state. However, such selective attention may totally ignore relationships between a problem state, its particular category of problem states, and

the particular operators. Thus, when problem solvers devote full attention to a goal, they may exclude those features of a problem necessary for learning.

Unlike a means-ends strategy, a *history-cued* strategy works with goal-specific and non-goal-specific problems (e.g., concept-discovery tasks). It uses actions from previous problem-solving episodes to generate actions for the current problem-solving episode. Sweller, Mawer, and Howe (1982) propose that a history-cued strategy makes use of rule induction, a primary means of knowledge transfer. Thus, a history-cued strategy could facilitate problem solving as well as learning.

A series of studies examined the issue of knowledge transfer using goal-specific problems solvable by either strategy: a rule-induction procedure (i.e., history-cued) or a non-rule-induction procedure (i.e., means-ends) [Sweller, Mawer, & Howe, 1982]. In situations where subjects chose a means-ends strategy, there was a comparative lack of transfer due to a failure to induce a rule. On the other hand, when subjects used a history-cued strategy, large performance changes on transfer tasks were observed. Insofar as this transfer indicated expertise or knowledge of a problem structure, use of a history-cued strategy resulted in more rapid acquisition of knowledge. Similar results were obtained in experiments where problem-solvers were presented with problems that had been modified to eliminate specific goals; problem-solving expertise was enhanced more rapidly (experiment referenced in [Sweller, 1988]).

These findings suggest that a history-based, user support tool may encourage the use of a history-cued problem-solving strategy rather than a means-ends strategy and thus, facilitate rule induction and knowledge transfer. Furthermore, such a tool may indirectly result in more rapid acquisition of knowledge.

History Information Helps Users Cope with Improvisation

A history contains knowledge and actions relevant to the particulars of the current situation (e.g., what actions were taken to arrive at the current situation). It also contains information about a user's evolving plan of action. However, unlike the plans that users have in their heads, user histories reflect improvisations made to their mental plans in order to fit the current situation; user histories are plans that are concrete, externalized, and directly accessible. Thus, they can help users figure out how they dealt with uncertainties in the past by relating the knowledge and actions needed during periods of improvisation. The relationship between a user history and plans-as-communication [1] and the ways in which a history can be enriched to support plans-as-communication are unexplored.

[1] Plans-as-communication relate the knowledge and actions needed during periods of improvisation (i.e., situated actions) [Agre & Chapman, 1990]. A user history is an example of plans-as-communication resource which helps users figure out how they dealt with uncertainties in the past.

History Tools Relieve Cognitive and Physical Burdens

User biases for the path of least cognitive resistance and repetition in user behaviour suggest strong user preferences for what has worked in the past, even if it is not an optimal approach. Users prefer to concentrate on conceptual difficulties associated with task execution rather than the tedious, mundane, and operational aspects of task execution. By reusing past approaches, users can rapidly repeat actions that are unchanged in the current task. Furthermore, users can benefit from improvements in their interactions as limited resources are more accessible and more effectively used.

Problem Statement

My thesis is that history, and tools based on history, can enhance user support. However, despite the potential benefits that both offer, very little attention has been focused on either and on their prospects for user support. In fact, there is little empirical evidence to illustrate the effectiveness of history tools and few studies and design efforts concerning history-based user support.

This dissertation investigates, through a series of studies, a number of specific questions related to the composition of a history and the prospects of history tools for user support. The dissertation explores deficiencies, potentials, and constraints of history and history tools. Each study poses specific questions concerning the effectiveness of history tools for user support. The studies examine the usefulness of current history tools, the extent of history-tool usage, the user interaction behaviours that can benefit from history tools, the ways that history tools can exploit such user behaviours, and the mental and physical costs associated with using history tools.

Dissertation Roadmap

The dissertation is divided into three parts. The first part, Chapters 2 and 3, characterizes the uses of current history tools and proposes requirements and options for enhancing the design of history tools. The second part, Chapters 4 and 5, describes an empirical observation and analysis of actual, everyday, natural user interactions with UNIX and UNIX history tools. The third part, Chapter 6, examines the effort involved in issuing repetitive user actions.

Chapter 2 focuses on how current history-like tools can enhance user interactions as well as the current state of this support. Various basic uses of a user history and their manifestations are characterized. System capabilities provided to support these uses are critiqued. Then, a number of existing systems are assessed in terms of the degree to which they support and integrate these uses.

Chapter 3 examines the needs and requirements for history tools which integrate the various uses of history. The kinds of history information and the kinds of functionalities that a history tool should support are proposed.

Chapter 4 is concerned with user interactions supported by history tools and user interactions with history tool usage. The findings from an exploratory study conducted with these concerns in mind are presented.

Chapter 5 examines in a formal way the user behaviour – user repetition of user actions – found in Chapter 4 and Greenberg and Witten (1988b)'s study. This behaviour is akin to behaviour observed in a program's references to computer memory. We adopt the concept that describes such program reference behaviours as a characterization of the repetition of user actions in our analysis of user interactions within a UNIX environment. Findings from a number of studies examining questions concerning this user behaviour and its implications for history-based, user support tools are presented.

Chapter 6 presents an analysis and comparison of the human information processes involved in issuing a recurrent command. The mental and physical effort involved in typing the recurrent command is compared to the mental and physical effort involved in using history tools. The comparative measures are the number and variety of different human information processing operations involved in using each proposed design. Since our comparative analysis relies on the predictions generated by cognitive models, it is important that these predictions are accurate. We conclude by examining the accuracy of these predictions.

Chapter 7 summarizes the dissertation and our research contributions. A number of future research directions for history as a user support tool are discussed.