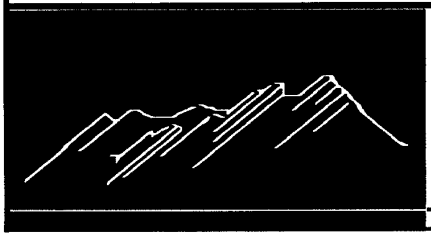
INSTITUTE OF COGNITIVE SCIENCE



Technical Report

University of Colorado, Boulder

A Comprehension-Based Model of Correct Performance and Errors in Skilled, Display-Based Human-Computer Interaction

Muneo Kitajima & Peter G. Polson

Institute of Cognitive Science
University of Colorado
Boulder, Colorado 80309-0344

Technical Report #94-02

A Comprehension-Based Model Of Correct Performance and Errors in Skilled, Display-Based Human-Computer Interaction

Muneo Kitajima
National Institute of
Bioscience and Human-Technology
1-1 Higashi
Tsukuba Ibaraki 305, Japan
+81 298 54 6731
kitajima@nibh.go.jp

Peter G. Polson
University of Colorado

Institute of Cognitive Science
Boulder, Colorado 80309-0345
+1 (303)492-5622
ppolson@clipr.colorado.edu

ABSTRACT
This paper describes a computational model of skilled use of a graphical user interface and of errors made by skilled users. The model is based of Norman's action framework and is implemented using Kintsch's construction-integration theory. The model selects actions, such as moving mouse cursor, clicking mouse button, typing letters, and so on, by integrating information from various sources. These sources include the display, task goals, expected device states, and knowledge about the interface and the application domain. We show how information about the intermediate states of a task presented on a display plays a critical role in skilled performance. The model also provides a well-motivated account of one kind of error, action slips, made by expert users.

1. INTRODUCTION

The goal of this paper is to present a computationally-based, performance model of skilled use of applications with graphical user interfaces like those of the Apple Macintosh and Microsoft Windows that accounts for both correct performance and errors made by expert users. The model provides a theoretical analysis of attributes of skilled human-computer interaction showing in detail how information about the intermediate states of a task presented on the display plays a critical role in performance. Our model is synthetic in that it attempts to integrate the views of numerous workers on the nature of graphically-based human-computer interaction (Smith, Irby, Kimball, Verplank, and Harslem, 1982; Shneiderman, 1982; Hutchins, Hollan, and Norman, 1986), theoretical ideas about the nature of display-based problem-solving (Larkin and Simon, 1987; Larkin, 1989; Howes, 1993), action planning (Mannes and Kintsch, 1991), and task and device representations (Payne, Squibb, and Howes, 1990).

Our results make two important contributions. First, the model provides a well-motivated explanation of the fact that skilled users make surprising numbers of errors (Card, Moran, and Newell, 1983; Norman, 1981; Reason, 1990; Hanson, Kraut, and Farber, 1984). Second, the model incorporates representations of large displays in which there is irrelevant information that the model must ignore in order to successfully complete a task. Thus, the model incorporates processes that focus on task-relevant information presented on the display and stored in long-term memory. This paper summarizes the results from four sets of simulation experiments that explore the parameter space of the model, validate the sufficiency of the model for a realistically complex task, and demonstrate how skilled users make errors.

1.1 How Does A Graphical User Interface Facilitate Performance?

Over the years, developers and designers (Smith et al., 1982; Bewley, Roberts, Schroit, and Verplank, 1983) have provided explicit rationale for graphical user interfaces. Shneiderman (1982) defined the concept of direct manipulation and argued that it is a critical property of successful graphical user interfaces. Hutchins, et al. (1986) developed a qualitative psychological model of interaction with a graphical user interface and provided a more detailed analysis of Shneiderman's concept of direct manipulation.

The key idea from Hutchins, et al. (1986) is that interaction with a system involves a cyclic process that has two major components: comprehension of the consequences of an action and planning an appropriate next action. The complexity of these comprehension and planning activities determines the difficulty of learning and performing a task.

1.2 Display-Based Problem-Solving

Larkin and Simon (1987) argued that displays facilitate problem-solving by allowing users to substitute perceptual operations for effortful symbolic operations, and that displays can reduce the amount of time spent searching for critical information. Larkin (1989) extended this analysis to tasks with characteristics similar to tasks performed using a computer. The first task she analyzed involved preparing fresh ground beans and assembling a coffee maker to brew coffee. The second task was manipulation of complex algebraic expressions to solve linear equations.

Classical models (Newell and Simon, 1972; Card, et al., 1983) assume that such tasks involve the generation or retrieval of hierarchical goal structures. This goal structure is an "isomorph" of the task structure that generated and held in working memory or represented in a set of complex plans that have been acquired and stored in long-term memory. Larkin (1989) argued that our subjective experience in actually performing one of these tasks calls into serious question the existence of these complex goal structures.

She identified the following six features of display-based problem solving for skilled users: 1) the process is easy, 2) it is largely error-free, 3) it is not degraded by interruption, 4) the steps are performed in a variety of orders, 5) the process is easily modified, and 6) performing the task smoothly and easily requires learning. She showed that rules that correctly interpreted representations of intermediate states of the problem presented on a display enable a user to generate the information contained in a complex goal structure. The user can read off a properly designed display information necessary to correctly select a next action. Howes (1993) defines such models as examples of recognition-based problem-solving architectures.

Numerous other authors with various theoretical motivations (*e.g.*, Suchman, 1987; Mayes, Draper, McGregor, and Oatley, 1988; Payne, 1991) have called into question classical assumptions about goal structures on the grounds that it is often difficult if not impossible to find any direct evidence for their existence. Larkin's results suggest that a well-designed graphical interface eliminates the need for the generation and maintenance of complex goal structures or the learning and storage of detailed action plans.

In the last several years, numerous researchers have developed models of display-based action planning (Chapman, 1987) and human-computer interaction (John and Vera, 1992; Peck and John, 1992; Howes and Young, 1991; Howes and Payne, 1990; Payne, 1991). They differ widely in the details of how they are implemented in an underlying cognitive architecture, e.g., SOAR (John and Vera, 1992; Peck and John, 1992; Howes and Young, 1991). However, they all are consistent with Larkin's (1989) argument that the control structure for a complex task can be read off a well designed display.

1.3 Errors in Human-Computer Interaction

A puzzling and frequently ignored fact in human-computer inetraction literature is that experts have surprisingly high error rates, up to 20%. The literature on errors has concluded that there are two qualitatively different types of errors (Norman, 1981; Reason, 1990). The first is errors of commission, or mistakes. Such errors are committed by users who are carrying out novel tasks and fail to immediately discover the correct action sequence. The other is slips, where expert users have the correct intention but fail to successfully execute the correct action sequence.

The following is a summary of representative studies of errors made by skilled users in human-computer interaction. Card, et al. (1983) studied individual skilled users performing two tasks, manuscript editing and electronic circuit design editing. The manuscript editing experiment involved a detailed evaluation of a single expert user doing 70 edits presented in marked up manuscript. Errors were made on 37% of the command sequences describing edits. Over half of the errors were detected and corrected during generation of the editing commands. Sixteen percent (15 out of 70) of the commands issued by this very skilled user generated the wrong result and required additional edits to correct these errors. In a second study of a single expert carrying out an electronic circuit design editing task, the user had an error rate of 14% on 106 edits.

Hanson, Kraut, and Farber (1987) studied 16 researchers and managers who were intermediate and expert level users of UNIX performing document preparation tasks and e-mail. They logged over 10,000 commands. The overall error rate was 10% with error rates ranging from 3% to 50% on different commands.

The experiments briefly reviewed here are representative of results from a wide range of studies

in the human-computer interaction literature. Error rates for expert users range from 5 to 20%. In all studies of experts, users eventually produced the correct results. Approximately 50% of the errors are detected during the generation of a command and corrected. Detection and correction of errors is an integral part of expert skill.

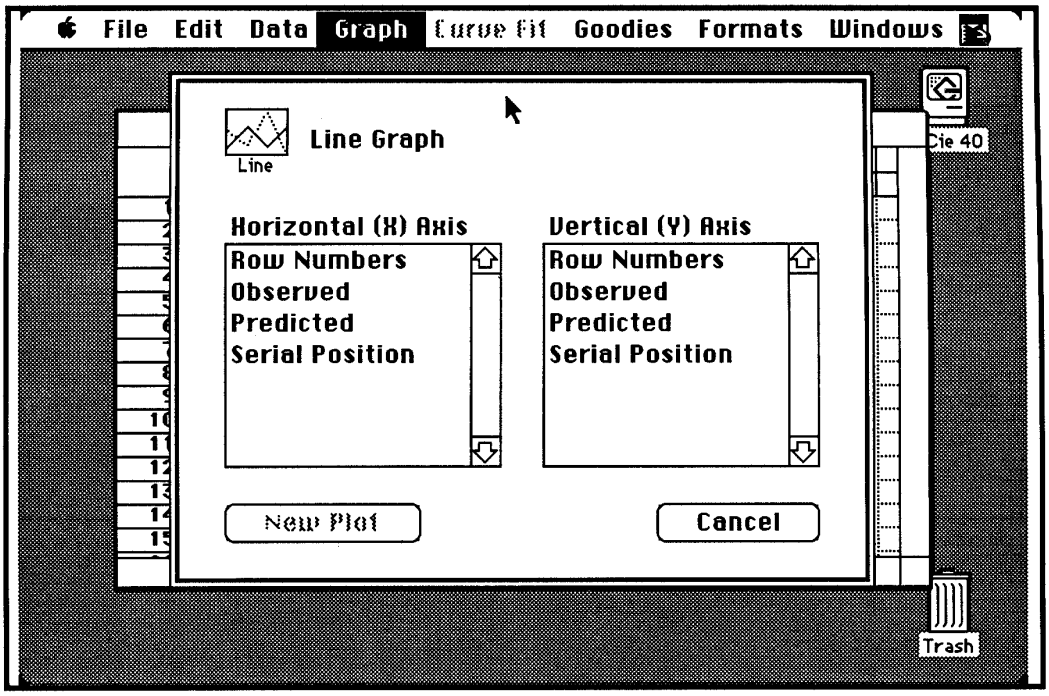


Figure 1. The critical intermediate state of the first subtask in the Cricket Graph Task. User must select "Serial Position" from the left scrolling window, "Observed" from the right, and then click "New Plot."

2. OUTLINE OF THE MODEL - A COMPLETE ACTION CYCLE

2.1 An Example Task

The major task studied in our simulation experiments involved preparing a graph that matches an example using Cricket Graph 1.3[1]. Here, we briefly describe the task and summarize the subjects' representation of the action sequence necessary to accomplish it. Our simulation of this task is described in detail in Section 4.3.2.

We assume that the user is a skilled user of Cricket Graph and that he or she has been given the data to be plotted in a Cricket Graph document entitled "Example Data." Double-clicking "Example Data" causes the program to display a spreadsheet with three columns labeled "Observed," "Predicted," and "Serial Position." The user's task is to plot "Observed" as a function of "Serial Position" and then edit the resulting default graph so that it conforms to a model provided by the experimenter.

The user's first subtask, creating the default graph "Observed" plotted as a function of "Serial Position," involves selecting "Line-Graph" from the "Graph" pull-down menu which brings up a dialog box. The dialog box, shown in Figure 1, enables the user to designate the column labeled "Serial Position" as the X-axis and the column "Observed" as the Y-axis. Clicking a button labeled "New Plot" causes the default graph to be presented.

The second major component of the task involves a sequence of editing operations that change X- and Y-axis ranges, the font and size of X- and Y-axis, legends, title, and the like. These editing operations

[1]Copyright Cricket Software, 1986-89, Valley Stream Parkway, Malverin, PA. Out of date version. Current version is published by Computer Associates.

enable the user to transform the default graph into a graph that matches the appearance of the model.

2.2 Overview of the Model

Our theory elaborates Norman's (1986, 1988) action theory framework shown in Figure 2. The four basic components are: (1) goals representing what the user wants to accomplish which are a schematic outline of the action sequence that will accomplish the task, (2) a task environment which is the world that reacts to the user's actions and generates new responses by modifying the display, (3) the stage of evaluation comprised of the processes that evaluate and interpret the display, and (4) the stage of execution comprised of the processes that select and execute actions that affect the world. Our model is mapped onto Norman's action cycle which assumes three processes for the stage of evaluation and three for the stage of execution.

We assume that the last action has led to a major change in the state of the display. In this case, the complete action cycle involves all six subprocesses shown in Figure 2. In the following sections, we briefly describing each subprocess in Figure 2.

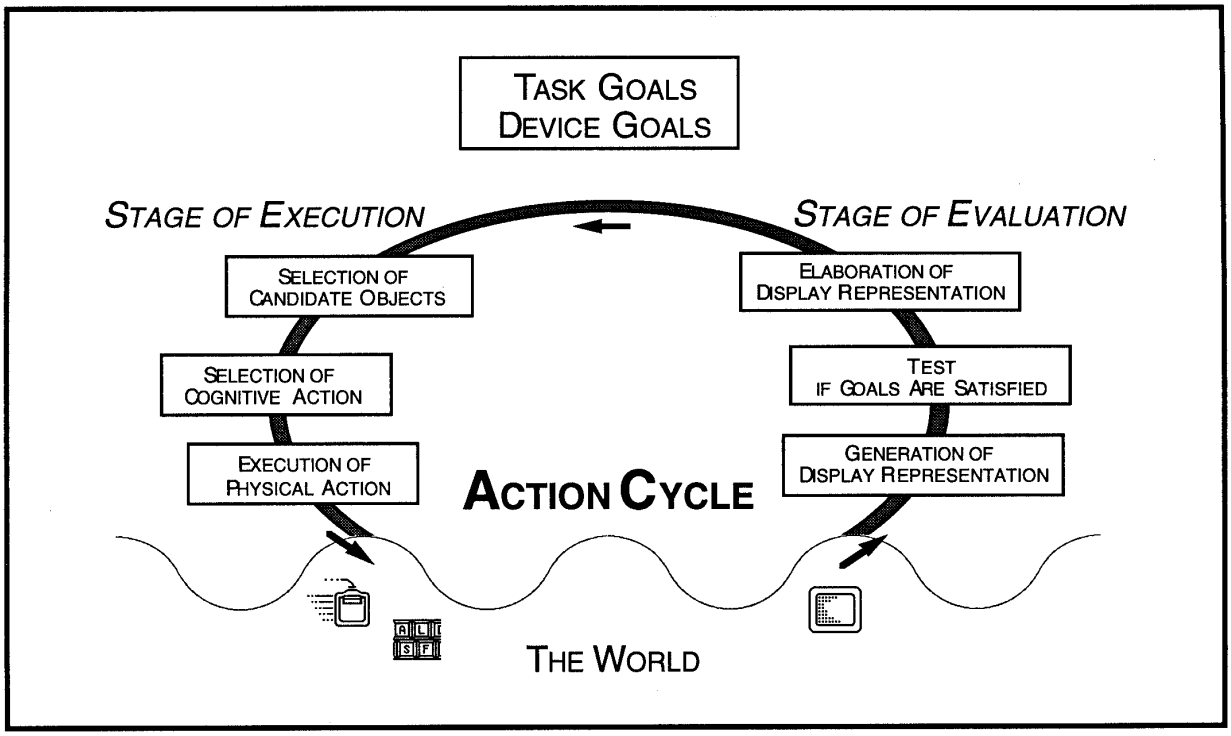


Figure 2. Overview of Norman's action cycle, defined by four components; goals, the stage of evaluation, the stage of execution, and the world.

2.2.1 Task Goals and Device Goals

The model assumes that skilled users have a schematic representation of the task and action sequence necessary to complete the task that is in the form of a hierarchical structure involving two kinds of goals: task goals and device goals. Our goal representation is taken directly from the Yoked State Space Hypothesis proposed by Payne, et al. (1990). Payne, et al. assume that discovering how to carry out a task involves searching of two problem spaces. The first is a space of possible task states. The second is a space of possible device states that are required to achieve a given task state. We assume that each task goal is associated with one or more device goals. The device goals specify device states that must be achieved in order to satisfy an associated task goal.

For the Cricket Graph Task, a skilled user's initial goal is to produce the default graph with "Observed" plotted as a function of "Serial Position" . A key device goal is the appearance of the dialog box that enables them to select the columns of the spreadsheet that will be plotted on the X- and Y-axis, respectively (Figure 1). Complete list of the task goals and device

goals for the Cricket Graph Task is presented in Table 2 in Section 4.3.2.

Our model simulates a skilled user who has complete and correct knowledge of the goal structure of the Cricket Graph Task. The model is capable of simulating errors even though we assume that the user has the correct task and device goals for each step.

2.2.2 Stage of Evaluation

There are three separate processes involved in the evaluation stage. (See the right portion of Figure 2.) The first involves generation of the display representation. The second checks to see if the current device or task goal has been satisfied. The third involves interpreting the meaning of information presented on the display via an elaboration process. The following sections describe these processes.

2.2.2.1 Generation of the Display Representation

The model assumes that the visual image of the screen is parsed into a collection of objects, each represented by several propositions. The parsing process is not implemented in the model.

The representation of each object on the display includes a limited amount of appearance information and no information about relationships to other objects on the display or the function of an object. Appearance information included in the representation of an object reflects the current system state. For example, the mouse pointer in the Macintosh interface changes in appearance depending on both the type and the current state of the object being currently pointed at. The pointer takes on an arrow shape on the desktop and an I-beam shape when over a selected, text object that is editable.

2.2.2.2 Goal Satisfaction

Associated with each task goal is a sequence of one or more device goals. The display representation is used to test whether or not the current device goal is satisfied. When a device goal is satisfied, it is replaced with the next device goal in the sequence associated with the current task goal. If it is the last device goal associated with the current task goal, the task goal is replaced with the next task goal.

2.2.2.3 Elaboration of the Display Representation

Interpreting the display is modeled in part by a *memory sampling process* that retrieves information from long-term memory related to the representation of the current display and task and device goals. The retrieved information elaborates the display representation, providing information about interrelationships between display objects, relationships between the task and display objects, and other attributes of display objects. The elaboration process simulates comprehension of the display that results from the last action in the context of the current task and device goals.

The elaboration process is probabilistic. As a result, there may be missing information that is necessary to properly interpret the display. The incomplete, elaborated display representation can cause the model to make an incorrect action. The evaluation process can fail even when an expert user has in long-term memory all of the information necessary to correctly interpret the display. Thus, the model can account for errors made by expert users. This process is described in Section 3.3.1.

2.2.3 Stage of Execution

The stage of execution involves three processes. These processes are shown in the left portion of Figure 2. The first involves the selection of a small number of candidate objects from the large number of display objects currently on the screen. This process is described in more detail in Section 3.2.2. The model then considers all logically possible actions that could be carried out on the candidate objects. The action selection process, described in Section 3.2.3, selects an action on one of the candidate objects and executes it. The model represents actions at a small grain size: individual mouse movements, click and double-click on pointed-at objects, drag, and the like. The model then retrieves a representation of the next screen that results from the selected action.

3. DETAILED DESCRIPTION OF THE MODEL

The model described in Section 2 has been simulated by a computer program based on Mannes and Kintsch's (1991) model of action planning derived from Kintsch's (1988) construction-integration theory of text comprehension. This section gives a brief review of the theory in the context of human-computer interaction. Next, we present detailed explanations of the representations used in the model, the memory sampling process, and the construction and integration processes.

3.1 The Construction-Integration Theory of Text Comprehension

Kintsch (1988) proposed a model of text comprehension that combines elements of both symbolic and connectionist models of cognitive processes. Kintsch's theory views text comprehension

as a cyclic process where a reader processes a sentence or the major constituent of a longer sentence during a single construction-integration cycle; comprehension of a text involves a sequence of such cycles. On each cycle, the model takes as input a representation of the reader's intentions, key elements of the text comprehended so far, and a propositional representation of the next sentence or major sentence fragment. The model outputs a representation of this latest sentence or fragment consistent with the reader's goals and the context provided by the previous text.

3.1.1 Text Comprehension Process

Kintsch (1988) assumes that the construction-integration cycle is a two-phase process. In the first phase, a network of propositions is created containing possible alternative meanings of the current sentence or fragment. The construction process generates an associative network whose nodes are propositions representing the input text, the meanings of words in the input text retrieved from long-term memory, the current context, and the reader's goals. Construction is a bottom-up process that is not guided by context. Thus, in elaborating the meanings of concepts contained in the representation of the current sentence, the construction process may create inconsistent representations.

The integration process, the second phase, is used to compute an interpretation of the input sentence consistent with the current context and the reader's goals. The integration process is connectionist in nature and uses a spreading activation mechanism. The most highly activated nodes in the network represent an interpretation of the input sentence that is consistent with the reader's goals and the current context.

3.1.2 Extensions to Human-Computer Interaction

Mannes and Kintsch (1991) extended the construction-integration theory to action planning. Their experimental task domain was human-computer interaction. Mannes and Kintsch's (1991) model took as input a representation of the user's or planner's goals, the text containing the task description, and a very schematic representation of the task context. They argued that comprehension and action planning can be conceived of as similar tasks. Readers and planners must integrate their goals and information from other diverse sources to select one out of numerous alternative interpretations of a text or one out of numerous competing plans for action. Mannes and Kintsch (1991) also noted that many human-computer interaction tasks and other action planning tasks are initiated by a request to a user or planner that is in the form of text. A natural extension of a model of comprehension is to show that it demonstrates its understanding by executing actions necessary to comply with a request contained in a text.

A new construct that they added to Kintsch's (1988) construction-integration model was the plan element. The plan element is a representation of a single action. In their case it was a high level action, like entering an operating system command or typing specified text. A plan element contains the following components: 1) one proposition that describes the consequences that would be obtained by executing it, 2) conditions that have to be true in the current context before the plan element can be executed, and 3) several propositions representing the detailed changes in the display caused by actually executing it.

Doane, Mannes, Kintsch, and Polson (1992a) and Doane, McNamara, Kintsch, Polson, and Clawson (1992b) extended this model to account for the behavior of expert users of UNIX who are able to combine elementary UNIX commands into complex interrelated sequences of actions that accomplish a novel goal.

3.1.3 A Process Model of Display-Based Human-Computer Interaction

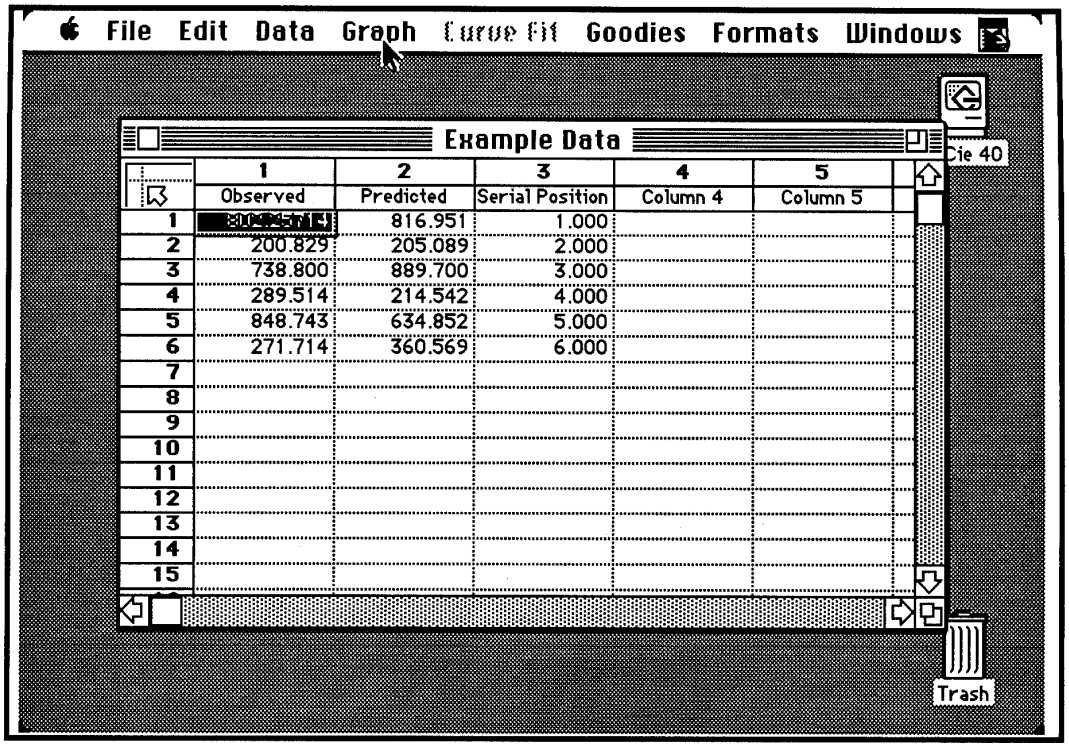
Kitajima and Polson (1992) and this paper develop a model of display-based human-computer interaction based on Mannes and Kintsch's (1991) construction-integration model of action planning. Our model extends previous models of human-computer interaction using the construction-integration framework in four ways.

Our first and most important extension is to show that the construction-integration model of action planning can provide a principled explanation of errors made by expert users.

Second, the model has a more detailed representation of information contained in realistically complicated displays. Thus, there are a large number of possible objects in the world that are candidates for action. This led to our third modification of the original framework. We have incorporated an explicit process in the model that selects candidate objects for possible actions. Generating an action involves two construction-integration cycles. In the first cycle, candidate objects consistent with the user's current goals and state of the display are selected for possible actions. During the second cycle, all possible actions

on the selected objects are considered, and one action is chosen that is consistent with a user's goals and the current display state.

Our fourth major extension to Mannes and Kintsch (1991) concerns our assumptions about the grain size of action. Both Mannes and Kintsch (1991) and Doane, et al. (1992a, 1992b) assumed large grain size actions, e.g. execution of complete commands. Kitajima and Polson (1992) and the model described in this paper assume a much smaller grain size, individual mouse cursor movements, click, double click, and hold. Our model contains no representation of sequences of actions like "select item X from menu Y." The model computes such action sequences based on the user's goals and the changing state of the display.



	1	2	3	4	5
	Observed	Predicted	Serial Position	Column 4	Column 5
1	[illegible]	816.951	1.000		
2	200.829	205.089	2.000		
3	738.800	889.700	3.000		
4	289.514	214.542	4.000		
5	848.743	634.852	5.000		
6	271.714	360.569	6.000		
7					
8					
9					
10					
11					
12					
13					
14					
15					

Figure 3. This is the state of the display just after the user has pointed at "Graph" menu item.

3.2 The Network Representation

3.2.1 The Display, Goals, and Information in Long-Term Memory

The display, task and device goals, and information in long-term memory are represented as propositions. We have adapted Bovair and Kieras's (1985) version of propositional notation for our purposes.

A proposition is a tuple of the form,

(predicate $argument_1$ $argument_2$ ... $argument_n$).

For example, the plain English version of one proposition from the display representation,

OBJECT23 is_a_kind_of DISPLAY-OBJECT

is formally represented as,

(is_a_kind_of OBJECT23 DISPLAY-OBJECT)

Notationally, the predicate of a proposition is in lower case letters connected by underscore characters (e.g., is_a_kind_of) and arguments, in small caps letters (e.g., OBJECT23).

3.2.1.1 Display Representations

The display is represented as a collection of display objects. Each display object is represented by six propositions in the current implementation. Recall that the representation of each display object contains a limited amount of information about the appearance of the object, and no information about semantics, legal actions, or relationships between objects.

Consider the menu items shown in Figure 3. Each item is a display object, and is represented by propositions identifying it and defining its display status. For example, **Graph** in the menu bar is represented with three propositions for its identification:

OBJECT23 is_on_screen, ...(P1)

OBJECT23 is_a_kind_of DISPLAY-OBJECT, ...(P2)

OBJECT23 is_a_kind_of GRAPH-MENU-ITEM. .....(P3)

and three proposition for its display status:

OBJECT23 is_pointed_at, ...(P4)

OBJECT23 is_not_highlighted, ...(P5)

OBJECT23 is_not_grabbed. .....(P6)

OBJECT23 is an arbitrary internal identifier unique for the specific display object, **Graph**.

P1 states that the object exists and is on the screen. P2 classifies the object as a display object. DISPLAY-OBJECT represents a class of display objects that allow for certain kinds of physical actions. P3 is a type-token relationship identifying the object. GRAPH-MENU-ITEM represents a type that subsumes any display object that is displayed in a menu with the name of "Graph." Propositions P4 to P6 define the status of the display object.

Note that our model uses a very simplified representation of display objects. There is no information about color (except for highlighting) shape, size, location, adjacent objects, containment, or textural features like the text in an icon label, font, size, and so on. The limited amount of perceptual information in the current model is not a constraint imposed by the basic representational formalism but is a decision made by us.

3.2.1.2 Goal Representations

The model assumes that expert users have schematic representations of task goals in long-term memory. A task goal is the representation of a user's intentions to perform actions on objects (Kieras, 1988). For example, the task goal for the Cricket Graph Task is represented as follows:

(TaskGoal NEWPLOT LINE-GRAPH GRAPH OBSERVED-DATA SERIAL-POSITION)

The above task goal can be paraphrased as "Plot a new line graph with observed data plotted as a function of serial position."

A device goal is the representation of the consequences of an action or sequence of actions in terms of the appearance of one or more objects on the display. The model assumes that expert users have representations of device goals in long-term memory, and can associate a task goal with one or a number of device goals. One of the device goals associated with the above task goal, for example, is an encoding of the display shown by Figure 1. The device goal consists of a list of key features of that display which would include the fact that OBJECT23 is selected, and the fact that the display is associated with the concept of LINE-GRAPH in some way represented as follows:

(DeviceGoal OBJECT23 SELECT LINE-GRAPH)

Retrieval of task and device goals from long-term memory was not simulated. The model was given the correct goals for each step of the action sequence necessary to perform the task.

3.2.1.3 Long-Term Memory Representations

Propositions representing the contents of long-term memory contain additional information about display objects. This information is used to elaborate the display representation. Propositions describing the contents of long-term memory for the Cricket Graph Task were coded based on an analysis of graph drawing tasks and the manual for the basic operations on Macintosh. Propositions represent part-whole relationships, attributes of objects, and possible functions invoked by different actions on an object.

The following examples show how the objects are represented in the model:

Part-whole relationships

When an object has a component, it is propositionalized by using the has predicate:

MENU-BAR has GRAPH-MENU-ITEM

Attributes

When an object is subordinate to a higher concept, or has a certain kind of attribute, is associated with another object, or with an application name, they are propositionalized as follows, respectively:

DISPLAY-OBJECT includes GRAPH-MENU-ITEM,

OBJECT23 is_not_a_kind_of TEXT,

OBJECT23 is_associated_with OBJECT24,

```
GRAPH-TITLE is_associated_with
  APPLICATION21,
```

where APPLICATION21 represents EDIT.

Possible functions invoked by different actions on an object
The following is the representation of the fact that the action grab on **Graph** will show its pull-down menu.

```
OBJECT23 when_it_is_grabbed FUNCTION11,
```

where Function11 represents Show-Pull-Down-Menu.

We already have the display representations for OBJECT23, the **Graph** menu item, that appears in Figure 3 as P_1 through P_6. The following demonstrates how knowledge of that display object can be elaborated through retrieval from long-term memory. The propositions P_1 through P_6 are elaborated around the three arguments, OBJECT23, DISPLAY-OBJECT, and GRAPH-MENU-ITEM.

OBJECT23 can be elaborated as follows:

```
OBJECT23 when_it_is_grabbed FUNCTION11,
OBJECT23 is_associated_with GRAPHS,(P7)
OBJECT23 is_not_a_kind_of TEXT,
OBJECT23 is_associated_with SCATTER-
  GRAPH-MENU-ITEM,
OBJECT23 is_associated_with LINE-GRAPH-
  MENU-ITEM,
MENU-BAR has OBJECT23.
```

Similarly, DISPLAY-OBJECT and GRAPH-MENU-ITEM can be elaborated as follows:

```
DISPLAY-OBJECT includes ICON-LABEL,
DISPLAY-OBJECT includes EDIT-MENU-ITEM,
DISPLAY-OBJECT includes GRAPH-MENU-ITEM,
DISPLAY-OBJECT includes COLUMN-GRAPH,
DISPLAY-OBJECT includes TEXT-GRAPH,
MENU-BAR has GRAPH-MENU-ITEM,
DISPLAY-OBJECT includes GRAPH-MENU-ITEM.
```

The rest of propositions in the display representation are elaborated in exactly the same way as above. The resulting representation, *the elaborated display representation*, defines the whole set of information that is associated with the current particular display state.

Table 1 shows the contents of long-term memory that were used for the simulation of the Cricket Graph Task. For the descriptive purpose, long-term memory is divided into nine domains. The table presents the numbers of propositions in each domain.

The Mouse domain, for example, includes propositions that define shape information of the mouse cursor. The other domains represent general facts about display objects like graphs, text, and spreadsheets. For example, the Graph domain includes propositions as follows:

```
GRAPH has GRAPH-TITLE,
LINE-GRAPH has LINE.
```

The Application domain represents knowledge about application such as,

```
CRICKET-GRAPH is_a_kind_of APPLICATION.
```

Table 1: Propositional contents of long-term memory

Domain	Number of Propositions
Dialog Box	46
Icon	34
Windows, etc.	28
Menu	23
Mouse	10
Graph	64
Text	48
Spreadsheet	12
Applications	7

3.2.2 Object Nodes and Object Selection

3.2.2.1 Representation of Object Nodes

Objects must be selected before any action can be performed on them. These objects must be selected from the elaborated display representation since it defines the context where the next actions are determined. The model searches for tokens representing display objects and generates corresponding propositions that state that an object is a candidate for action.

3.2.2.2 Candidate Object Selection

The total number of object propositions is the number of objects on the screen. For example, in Figure 3, there are ten objects that represent the menu bar. In addition, there are objects defined by elements of the

window, icons, and the like. In order to select three candidate objects, the model first concatenates object propositions with the elaborated display representation along with the current task goals and device goals. Then, it estimates the degree of relevance of each object to the task and device goals.

The degree of relevance is calculated by mechanisms defined by the construction-integration theory. The construction process links nodes representing the task and the device goals, the objects displayed on the screen, the propositions retrieved from long-term memory by the elaboration process and the nodes representing object propositions. The integration process distributes activation over the nodes in the network where goal and display nodes are sources of activation. Finally, the model selects the three highest activated object nodes as objects that will be candidates for action.

3.2.3 Action Representations and Action Selection

The model represents actions at a very small grain size, at the level of individual mouse cursor movements, single clicking, and the like.

The model represents actions in three different ways. The first is a representation at the level of physical operations. The other two are forms of cognitive representations of the physical actions. The first form is generic cognitive actions stored in long-term memory. The second is specific cognitive actions which are generated by specializing the generic cognitive actions to the current context. We describe them in the following sections.

3.2.3.1 Physical Actions

In the current implementation, the model simulates the following six physical actions:

Move Mouse Cursor,

Single Click,

Double Click,

Press and Hold Mouse Button Down,

Release Mouse Button,

Type.

These actions are sufficient to simulate the Cricket Graph Task.

3.2.3.2 Generic Cognitive Actions and Specific Cognitive Actions

Several generic cognitive actions are defined for each physical action. For example, two cognitive actions can be derived from *Move Mouse Cursor*. One cognitive action represents an action that generates the display shown in Figure 3 by moving the mouse cursor to **Graph** in the menu bar. Another points at the highlighted cell in the spreadsheet with the goal of editing it. The purpose of the former is to perform the function defined by **Graph**, SHOW-PULL-DOWN-MENU, and the purpose of the latter is to get an insertion point for editing the cell value. Cognitive actions are characterized both in terms of their purposes and the display objects they operate on.

The model assumes two forms of cognitive actions: specific cognitive actions and generic cognitive actions. Specific cognitive actions are those cognitive actions for which display states and purposes are fully specified, so that they can be executed if the current display satisfies the conditions for their evocation. Generic cognitive actions are those cognitive actions where the objects and/or the purposes are defined as variables. They can be regarded as templates for generating actions that are specific to the current display state. Generic cognitive actions are assumed to be stored in long-term memory.

In the action selection process, the representations for all generic cognitive actions are retrieved from long-term memory. A combinatorial binding process generates all possible specific cognitive actions defined by the values of candidate objects and purposes. This process occurs without any considerations of whether the resultant representations of specific cognitive actions can be executed in the current context.

In the Cricket Graph Task, the six physical actions are mapped onto 18 generic cognitive actions. These generic cognitive actions are defined by states of display objects such as whether the or not object is text , whether or not it is highlighted , and whether or not it is grabbed, etc. Different generic cognitive actions are defined by combinations of states of display objects and their purpose; e.g., to get an insertion point in a highlighted text object, to select any display object, to deselect any display object, to launch an application, and so on. Such combinations generate six generic cognitive actions for *Move Mouse Cursor*, three for *Single Click*, two for *Double Click*, three for *Press and Hold Mouse Button Down*, two for *Release*, and two for *Type*.

3.2.3.3 Representation of Cognitive Actions

Both generic and specific cognitive actions are represented in a single form which has four fields[2]:

```
(label
 (associated physical action followed
 by display features and purposes)
 (a set of conditions)
 (a set of consequences))
```

The first field holds a label for identification of the cognitive action. The second field is a list with the first argument being the associated physical action followed by several arguments that enumerate major features of the cognitive action in terms of its purposes and display states. The third field holds a set of conditions tested to determine executability of the cognitive action in the context defined by the current network. The fourth field holds a set of propositions that are added to the network representing the consequences of executing the cognitive action.

In the action selection process, representations of specific cognitive actions are concatenated with the elaborated display representation and the appropriateness of each specific cognitive action to the specific context is estimated.

Three aspects affect the estimation of the degree of appropriateness. First, the second field establishes links to the current context and gains activation from the current context. Second, the conditions and the consequences are used to establish causal relations between specific cognitive actions. These causal relations affect the process of activation. Third, the conditions are matched against the elaborated display representation. If all the propositions in the third field exist in the elaborated display representation, the set of conditions is satisfied, and the specific cognitive action can be executed. The model selects the most highly activated eligible action as the action to be executed.

3.2.4 Weights of Links

We have described how the model represents goals, knowledge stored in long-term memory, the states of display, and cognitive actions. All are represented as nodes and interconnected, if connection exists, by either excitatory or inhibitory links. The nodes and links define a network structure.

---

[2] The same construct is called a plan element in Mannes and Kintsch (1992)

The links and their weights are established by the simulation program. Different types of links have different strength defined by the following parameters:

W_{arg}:	argument overlap weight,
W_{assoc}:	free association weight,
$W_{action\text{-}e}$:	cognitive action excitatory weight,
$W_{action\text{-}i}$:	cognitive action inhibitory weight,
F_{task}:	task goal magnification factor, and
F_{device}:	device goal magnification factor.

In the following sections, details of the parameters will be described with their psychological implications. The parameters, except for the task goal and the device goal magnification factors, are the same as those used in Mannes and Kintsch (1991).

3.2.4.1 Argument Overlap Weight: W_{arg}

In the construction-integration theory, nodes in the network representing propositions are associatively related to each other by links. The number of shared arguments, *argument overlap* , determines the strength of a link between a pair of nodes.

For example, a display representation,

```
OBJECT23 is_on_screen,
```

and a representation in long-term memory,

```
OBJECT23 is_associated_with LINE-GRAPH-
  MENU-ITEM,
```

are connected by the argument overlap mechanism by the shared argument, OBJECT23.

It is assumed that, when two nodes share one argument, they are connected by a link of strength W_{arg}. When they share N arguments, the strength is multiplied by N. This link is symmetric.

3.2.4.2 Free Association Weight: W_{assoc}

When an argument in a proposition representing a task goal, a device goal, or display object is successfully used as a retrieval cue for a proposition in long-term memory, the strength of the link(s) between the proposition containing the cue and the retrieved proposition is increased by the value of the free association weight, W_{assoc}. For example, when P_1 retrieves P_7, the strength of the link between them becomes $W_{arg}+W_{assoc}$. This link is symmetric.

3.2.4.3 Action Excitatory/Inhibitory Weight : $W_{action\text{-}e}$ and $W_{action\text{-}i}$

The representation of each cognitive action includes a set of conditions and consequences. A special relationship between conditions of one cognitive action and consequences of another establishes causal relations. If a condition of cognitive action, L_i, is satisfied by execution of L_j, then L_i supports L_j. If a condition of L_i is disabled by the execution of L_k, then L_i inhibits L_k.

For example, let L_i be the cognitive action, "single click within a word to locate the insertion point." L_i supports any pointing action which results in an I-beam cursor; it inhibits any action that yields an arrow shaped cursor. The strengths of excitatory and inhibitory links are parameterized by $W_{action\text{-}e}$ and $W_{action\text{-}i}$, respectively. These causal links are asymmetric.

The simulation also uses inhibitory links to prevent repeated execution of a specific cognitive action. The program examines for each cognitive action whether all consequences are found in the nodes representing the elaborated display representation. If they are found, the cognitive action is inhibited by all nodes that match the representation of consequences. The strength of the inhibitory links is $W_{action\text{-}i}$. These links are asymmetric.

3.2.4.4 Task/Device Goal Magnification Factor: F_{task} and F_{device}

Links between arguments in propositions representing task and device goal and other propositions in the network have a special status. The strengths of these links are multiplied by the task goal magnification factor, F_{task}, and the device goal magnification factor, F_{device}, which are greater than or equal to 1. These special links have strong effects on the elaboration process, object selection, and action selection. These links are symmetric.

3.3 Memory Sampling, Construction and Integration Processes

3.3.1 Memory Sampling Process

Each argument in each proposition that represents a goal or display object can serve as a retrieval cue for propositions in long-term memory. The retrieval process model was first described by Raaijmaker and Shiffrin (1981). Each argument is used as a retrieval cue N_{sample} times, *the elaboration parameter*. The probability that proposition P_i will retrieve proposition P_j, $Prob\ (P_j|P_i)$, is given by the following formula:

$$Prob\ (P_j|P_i)=\frac{W_{P_i,P_j}}{\sum_{k\neq i} W_{P_i,P_k}},$$

where $W_{P_i,P_j}\geq 0$ is the strength between proposition nodes P_i and P_j, which is calculated by multiplying W_{arg} by the number of arguments shared by nodes P_i and P_j, and in case where proposition P_i is either a task goal or a device goal, the strength is further multiplied by F_{task} or F_{device}.

The magnification factors, F_{task} and F_{device}, and N_{sample} control the breadth of elaboration. Propositions required to make a correct action executable may not be retrieved if they do not have strong relationships with the retrieval cue and/or if N_{sample} is small.

Figure 4 gives a theoretical explanation of this mechanism. Suppose that there is a proposition representing a display object, P_{disp}, which has connections to a task goal, P_{task}, a device goal P_{device}, other display objects, $P_{disp'}$, $P_{disp''}$ and information in long-term memory, P_{LTM}, and $P_{LTM'}$, one of which is critical for a correct action to be executed because it is included in its conditions. And suppose that each pair has one overlapping argument. Then, the link weights are assigned as shown in the figure. The probabilities for each proposition to be associated with P_{disp} in the sampling process are calculated by normalizing the sum of link weights to be one, as shown in the bottom of the figure. The link weights between P_{disp}, and P_{task} and P_{device} are multiplied by F_{task} and F_{device}, respectively, which means that when they become larger, the proportion of probabilities distributed to propositions other than the goals become smaller. This effectively reduces the possibilities for propositions other than P_{task} and P_{device} to be associated with P_{disp}, and makes it more difficult for information in long-term memory to be retrieved if the elaboration parameter is the same.

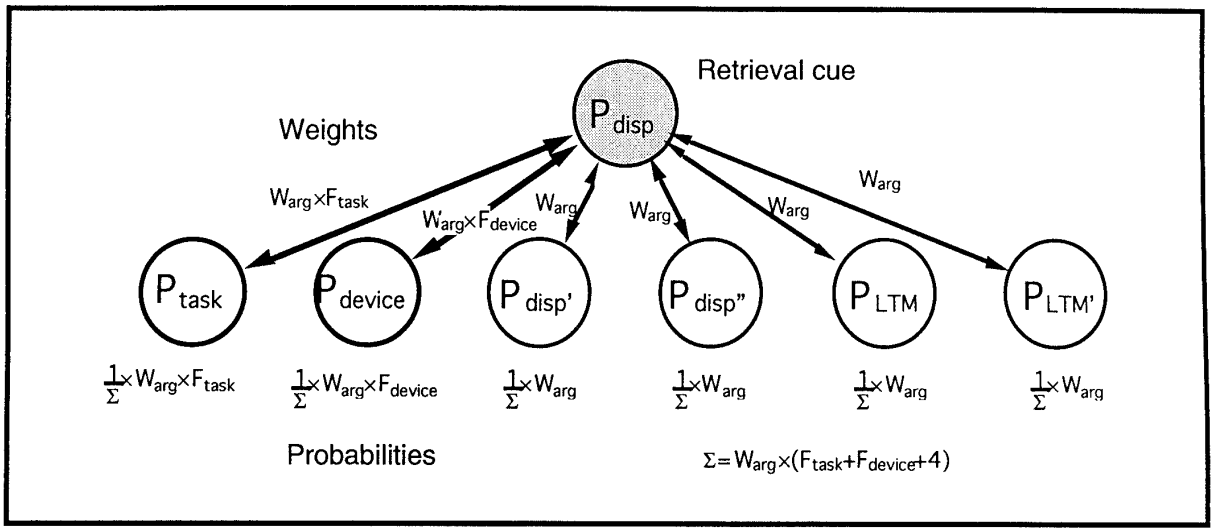


Figure 4. Explanation of mechanism of potential retrieval failure due to interactions between the goal magnification factor parameter, $F_{task,\ device}$, and the memory sampling process.

3.4 Summary of The Model

This section summarizes how the action cycle model is implemented using the construction-integration theory. To begin with, the model reads the current task and device goals and the display presentation for the current step from a file. The display representation is then elaborated by the memory sampling process . Each argument in each task, device, and display object proposition is used as a retrieval cue N_{sample} times.

In the object selection process, the model searches for tokens representing display objects and generates corresponding object propositions. The construction process establish links and assigns their weights between the task goals, the device goals, the elaborated display representation, and the object propositions. Then the network is integrated. The sources of activation are the propositions representing task goals, device goals, and display objects. Three object propositions with the largest activation values are selected as candidates objects.

In the action selection process, the generic cognitive actions are retrieved from long-term memory and their variables are bound to the specified candidates. In the construction process, the task goal, the device goal, the elaborated display representation, and the specific cognitive actions are linked. The network is then integrated and the most highly activated and eligible action is selected as the next action.

Mathematical details of the construction process and the integration process are described in Appendix A.

4. SIMULATION EXPERIMENTS

4.1 Outline of Simulation Experiments

The following sections describe simulation experiments that were performed (1) to determine appropriate parameter values for the link weights and (2) N_{sample}, (3) to demonstrate that the model can successfully perform tasks, and (4) to further understand the processes that generate correct and incorrect action sequences. The Icon Label Editing Task, requiring five steps, was used for the first two experiments, and a more complex task, the Cricket Graph Task requiring 12 steps, was simulated in the third and fourth experiments.

An experiment consisted of a set of simulation runs. Each simulation run generated the sequence of actions necessary to perform a task. In generating a single action, the elaboration process and the action selection process were simulated by a computer program. For the other processes involved in generating a single action, the necessary information was provided in input files read by the simulation. These files contained the assumed contents of long-term memory and display representations. We adapted a program, NETWORK (Mannes and Roushey, 1990) to carry out our simulations.

The primary purpose of the first experiment was to search the parameter space to find an appropriate

region of parameter values that characterizes expert performance. A large value of N_{sample} , the elaboration parameter, was used so that all relevant knowledge in long-term memory would be incorporated in the network with a high probability. In the second experiment, the value of N_{sample} was manipulated in order to study the functional relation between error rate and completeness of elaboration.

The purpose of the third experiment was to test the sufficiency of the model using a much more complex task, the Cricket Graph Task, using displays containing relevant and irrelevant display objects. In the fourth experiment, the values of N_{sample} were varied over the range from 4 to 24, manipulating the completeness of the elaboration process. Some steps with high error rates were studied in detail. In addition, recovery from a wrong action was simulated by extending the fourth experiment.

4.2 Expert Performance on A Simple Task

The first experiment explored the parameter space to find a set of parameter values that characterize expert performance. Expert performance by the simulation was defined as the generation of the correct action sequence and the rapid convergence of the integration process. The model contains seven parameters.

A subspace of the full parameter space was selected using results of pilot simulation experiments. The elaboration parameter, N_{sample}, was set to 14 assuring that all relevant knowledge was incorporated into the network. The free association weight, W_{assoc} was fixed to 1.0. The values of the task goal magnification factor, F_{task} and the device goal magnification factor, F_{device}, were set equal; the common value is referred to as the goal magnification factor, $F_{task, device}$.

The first experiment consisted of eighty one simulation runs each of which corresponded to a single point in the parameter space. The following values were assigned to each of the four parameters defining the subspace: W_{arg} (0.25, 1.0, 4.0), $W_{action\text{-}e}$ (0.25, 1.0, 4.0), $W_{action\text{-}i}$ (-0.25, -1.0, -4.0), and $F_{task, device}$ (1.0, 4.0, 16.0), where W_{arg} is the argument overlap weight, $W_{action\text{-}e}$ is the cognitive action excitatory weight, $W_{action\text{-}i}$ is the cognitive action inhibitory weight, and $F_{task, device}$ is the goal magnification factor.

4.2.1 Icon Label Editing Task

The Icon Label Editing Task involved editing the label of an document icon labeled "Example Data" by inserting text. There were just two icons on the simulated desktop: the document icon, "Example Data," and an application icon labeled "Cricket Graph." The correct sequence of actions was:

Step 1: move the cursor to the icon labeled Example Data,
Step 2: single click,
Step 3: move the cursor to insertion point in label field of the icon,
Step 4: single click obtaining the insertion cursor,
Step 5: type text.

The task goal given to the simulation was "to edit the label of the icon labeled Example Data by inserting text." The device goal for Steps 1 and 2 was "to see the icon Example Data highlighted." For Step 3, it was "to see the arrow cursor in the icon label field." For Step 4, the device goal was "to see insertion cursor there." For Step 5, it was "to see new text inserted." The simulation was provided with correct displays for all Steps regardless of the action selected by the model for that Step. Step 2 was an exception; the display generated by Step 1 was always used even if Step 1 was incorrect. The candidate object selection process was not simulated in Experiment 1; the model was provided with three candidate objects including the correct one on each Step.

4.2.2 Experiment 1 – Exploration of Parameter Space

4.2.2.1 Results

Figure 5 shows the results of the 81 simulation runs. The figure contains nine subgraphs. Each subgraph shows the outcomes of nine simulation experiments for a given combination of values for the goal magnification factor, $F_{task, device}$, and the argument overlap weight, W_{arg}. The nine cells in a given subgraph represent the outcomes of each of the runs defined by a combination of $W_{action\text{-}e}$ and $W_{action\text{-}i}$.

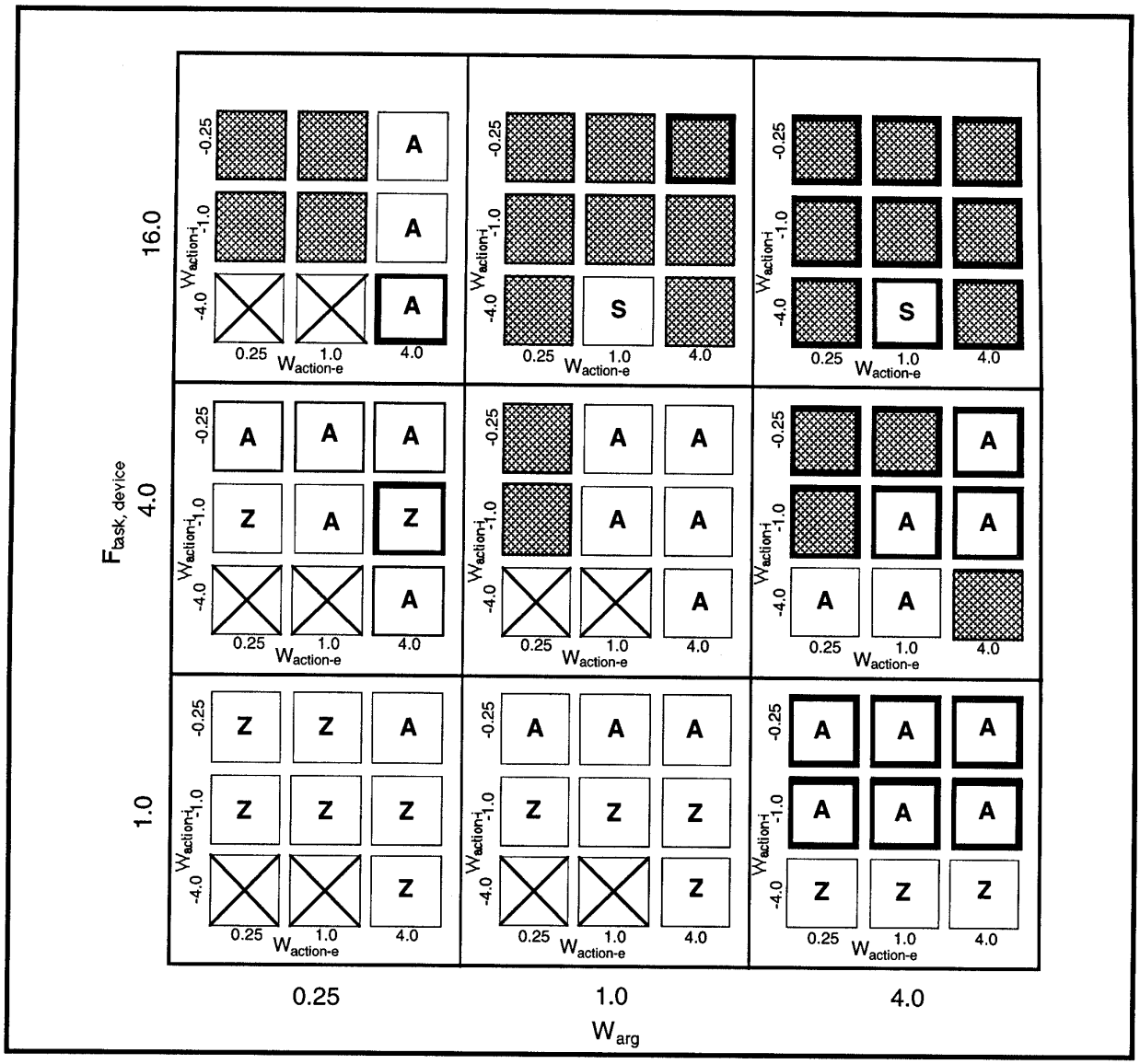


Figure 5. Results from the first experiment using Icon Label Editing Task. Each simulation run is defined by a combination of values assigned to four parameters. **S**, **A**, and **Z** stand for sampling error, activation error, and serious activation error, respectively. See the text for detail explanation.

The results of each run were classified in terms of correctness of the action sequence and speed of convergence of the integration process. A cross-hatched cell denotes a parameter combination generating the correct sequence; otherwise the stimulation made an error. Rapid convergence, 10 or less iterations of the integration process, is denoted by a bold boarder around the cell. A cross through a cell denotes a combination of parameter values where the integration process failed to converge, oscillating between zero and non-zero activation values for nodes of specific cognitive actions.

The letters **S**, **A**, and **Z** in a cell denote the outcome of an incorrect action sequence. **S** is a *sampling error* caused by failure of the elaboration process to incorporate all necessary information in the network. **A** is an *activation error* were the correct action did not get the highest activation value. **Z** denotes a cell were the correct action received an activation value of 0.0.

4.2.2.2 Expert Performance

Examination of Figure 5 shows that eight of nine simulation runs with W_{arg} = 4 and $F_{task,\ device}$ = 16 satisfied our definition of expert performance for the simulation. Furthermore, the results show that the goal magnification factor, $F_{task,\ device}$, is the major determinate of the probability of a correct action sequence. W_{arg} determines the speed with which the integration process converges. Large values of $W_{action\text{-}i}$, -4.0, can cause the integration process to oscillate.

4.2.2.3 Errors

The model makes errors for two reasons: failure to retrieve necessary information from long-term memory due to the elaboration process, and activation errors. This discussion focuses on activation errors. We explore retrieval failures, i.e. sampling errors, in Experiment 2.

An activation error occurs when the correct specific cognitive action does not get the highest activation value among executable actions during the integration process. This can occur even when the network contains representations of the conditions necessary to satisfy those of the correct cognitive action. These activation errors tended to occur for intermediate values of $F_{task,\ device}$ and W_{arg}. Activation errors are indicated by **A**'s in Figure 5. Some simulation runs suffered from serious activation errors, meaning that the activation values for some of the correct cognitive actions in the sequence were suppressed to zero; the corresponding cells are indicated by **Z**'s.

4.2.2.4 Conclusions

The simulation exhibits expert performance for large values of $F_{task,\ device}$ and W_{arg}. Large values of $F_{task,\ device}$ causes the elaboration process to focus information in long-term memory related to the task and device goals. In the integration process, large values of $F_{task,\ device}$ causes large activation flows from the goals to cognitive actions that have direct and/or indirect links to the goals.

In addition, the results are consistent with our intuition that higher strength links between nodes in the network cause the integration process to converge more rapidly making the action selection faster. The degree of relationships are expressed in terms of the value of W_{arg} defining the strengths of connections among propositions, and the values of $W_{action\text{-}i}$ and $W_{action\text{-}e}$ which link cognitive actions. Almost half the nodes in the network represent cognitive actions. Thus, $W_{action\text{-}i}$ and $W_{action\text{-}e}$ determine the strengths of a significant number of links in the network. W_{arg}, $W_{action\text{-}i}$ and $W_{action\text{-}e}$ determine the strengths of connections among propositions and thus the speed of converge of the integration process.

4.2.3 Experiment 2 - Errors on a Simple Task

For many steps in both the Icon Editing Task and the Cricket Graph Task, the display representation must be elaborated with the propositions retrieved from long-term memory in order for the model to be able to select the correct action for a given step. These propositions can link goals and display objects to the correct specific cognitive action.

Recall that the elaboration process is modeled by a random sampling process where retrieval probabilities are determined by the number of shared arguments, the goal magnification factor, $F_{task,\ device}$ and the number of times that each proposition in the goals and display is used as a retrieval cue, N_{sample}.

The goal of this experiment was to determine the probability of successfully completing the Icon Editing Task as a function of N_{sample}. Fifty simulation runs were carried out for N_{sample} = 4, 6, 8, 12, and 14, with the remaining parameters of the model set equal to the values that produced expert performance in Experiment 1, W_{arg}= 4, $F_{task,\ device}$= 16, $W_{action\text{-}e}$= 4, $W_{action\text{-}i}$= -4, and W_{assoc}= 1. The results are plotted in Figure 6 which shows that the probability of successfully completing the task ranges from . 22 (N_{sample}= 4) to . 98 (N_{sample}= 14).

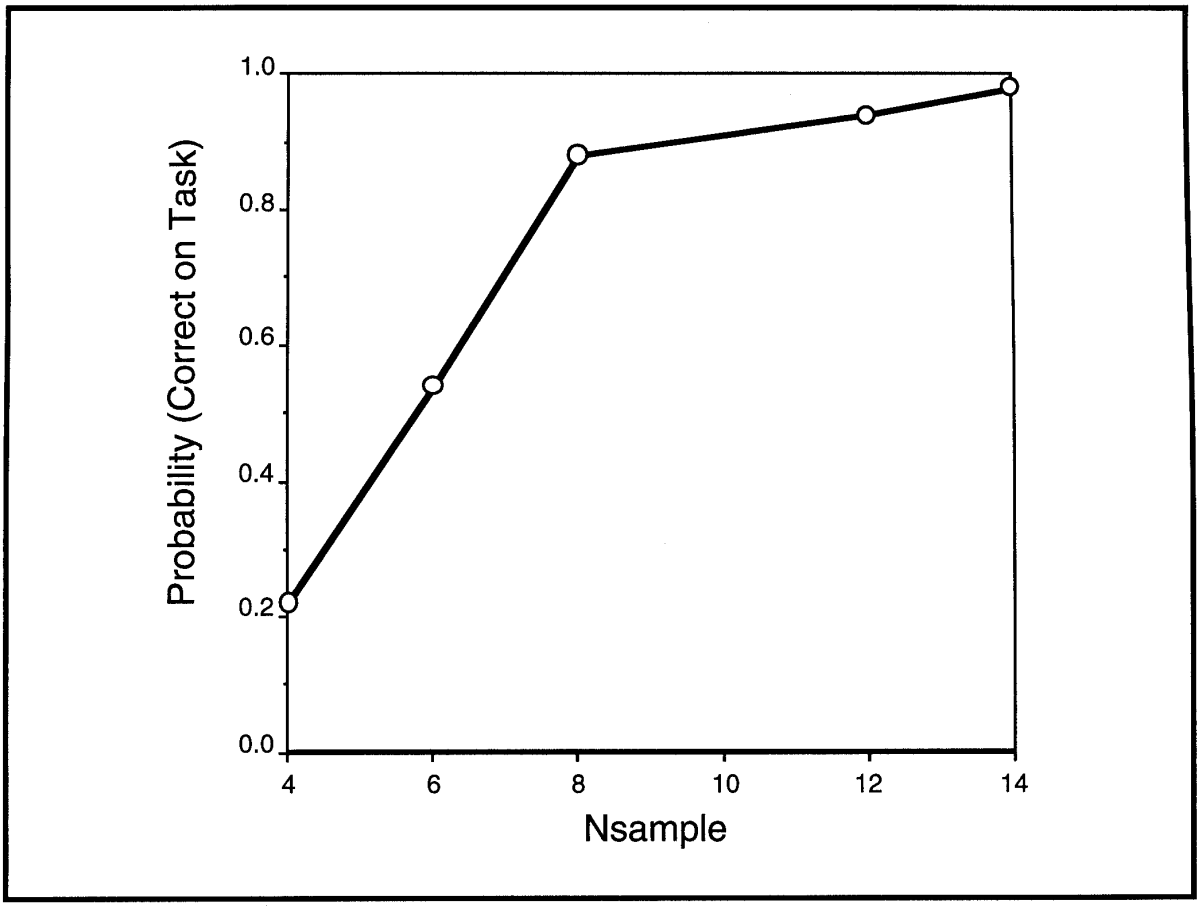


Figure 6. The probability of successfully performing Icon Label Editing Task as a function of the elaboration parameter, N_{sample}.

To the best of our knowledge, the results of this simulation experiment are the first detailed account of a well-known and puzzling result in the human-computer interaction literature, that experts have relative high error rates, up to 20% (e.g., Card, et al., 1983; Hanson, et al., 1984). The model is capable of simulating error rates in this range with values of $N_{sample} \approx 8$. Experiment 4 will further explore the details of the relationship between the elaboration process and error rates in a more complex task.

4.3 Expert Performance on A Realistically Complex Task

Experiment 3 demonstrates that the model can perform a realistically complex task where the display contains both relevant and irrelevant information. In addition, we show that the same values of the parameters used to simulate expert performance in Experiment 1 generalize to a much more complex task. Experiment 4 explores in much greater detail the determinates of error rates on different steps of a task. We will show that the model does not guess but selects the most reasonable action by making full use of the available information. Finally, we will demonstrate the model's ability to recover from errors.

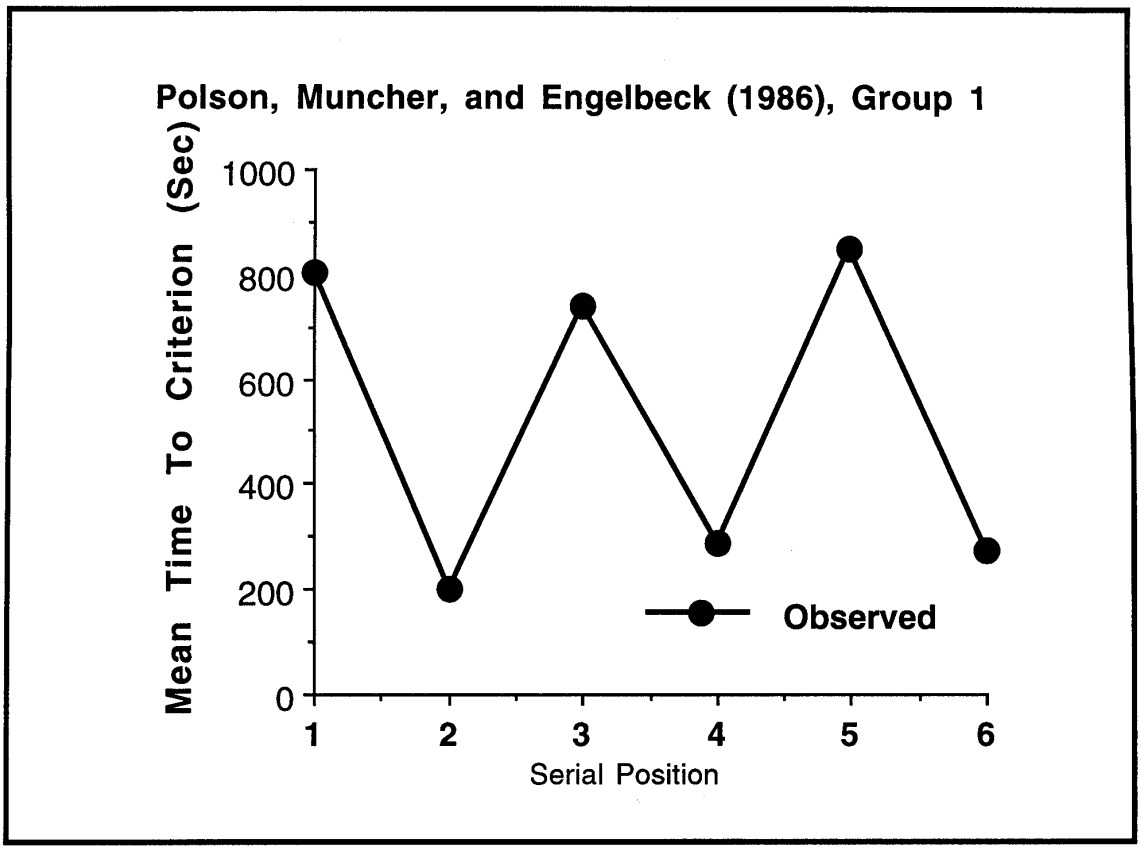


Figure 7. The graph to be produced by the user during the Cricket Graph Task.

4.3.1 Cricket Graph Task

The Cricket Graph Task involves the first two subtasks of the complete process of reproducing the graph shown in Figure 7 using Cricket Graph on a Macintosh computer. This task starts at the Cricket Graph application icon and the document icon for "Example Data" on desktop; double clicking on "Example Data" leads to Figure 3 with the mouse cursor on the background of the desktop. The first subtask is to plot the data in the column labeled "Observed" as a function of the column labeled "Serial Position." See Figure 3. The second subtask is to edit the graph title. Each of these subtasks is represented in the model as a task goal.

4.3.2 Experiment 3 - Test of Sufficiency of the Model

The task and device goals and correct action are shown for each of the 12 steps in Table 2. Ten steps are required to accomplish the first task goal, and two for the second. Four device goals are associated with the first task goal, and one for the second. The model was provided with the correct displays after the simulated action on Steps 1, 3, 4, 5, 7, 9, 11, and 12 regardless of the correctness of the previous action selection process. The model was also provided with the correct task and device goals.

The set of parameters that successfully simulated expert performance in Experiment 1 , W_{arg}= 4, $F_{task,\ device}$= 16, $W_{action\text{-}e}$= 4, $W_{action\text{-}i}$= -4, W_{assoc}= 1, and N_{sample} = 14, was used for this experiment. We recorded the propositions that were retrieved by the elaboration process on each step of the task and classified them by knowledge domain (See Table 1).

Table 2. Task goals, device goals, and correct actions for Cricket Graph Task

Step No.	Task (**TG**) and Device goals(**DG**)	Correct Action
	Subtask 1	
	TG-1: to create a default line graph with "Serial Position" as X axis versus "Observed" as Y axis	
1	**DG-11**: to see entry into the line graph environment	*Move Mouse Cursor* to **Graph**
2		*Press and Hold Mouse Button Down*
3		*Move Mouse Cursor* to **Line**
4		*Release Mouse Button*
5	**DG-12**: to see **Serial Position** is selected as X axis	*Move Mouse Cursor* to **Serial Position** in X axis selection list
6		*Single Click*
7	**DG-13**: to see **Observed** is selected as Y axis	*Move Mouse Cursor* to **Observed** in Y axis selection list
8		*Single Click*
9	**DG-14**: to see **New-Plot** is selected	*Move Mouse Cursor* to **New Plot**
10		*Single Click*
	Subtask 2	
	TG-2: to edit the graph title	
11	**DG-21**: to see entry into the editing environment	*Move Mouse Cursor* to **Graph-Title**
12		*Double Click*

4.3.2.1 Results

The correct action sequence was generated on the first simulation run. The average percentage of propositions retrieved from long-term memory ranged from 32% to 42% across Steps. For some domains, the proportion was relatively constant throughout the task: the Windows, Mouse, Icon, Text, and Spreadsheet domains . All propositions in the Spreadsheet domain, and almost all propositions in the Mouse (90%) and Windows (96–100%) domains were retrieved throughout the task. These results can be understood by considering what was displayed at each action selection. Display objects that are related to these domains appeared in the display throughout the task. On the contrary, few propositions were retrieved from the Icon domain (15–26%) and the Text domain (0–6%) which were not part of the task.

Figure 8 plots the proportion of propositions retrieved in each domain as a function of Step. The peak in the curve for the menu domain on Steps 3 and 4 is due to the appearance of the pull down menu. A dialog box was on the screen during Steps 5 through 10. More propositions were retrieved from the Graph domain when the default line graph appeared in Steps 11 and 12.

Figure 8 is a clear illustration of what we mean by display-based human-computer interaction. It shows how the relevant knowledge is retrieved from long-term memory based on the current contents of the display. The next experiment manipulates the values of the elaboration parameter showing how the model makes errors when the necessary information has not been retrieved from long-term memory. The causes of these errors will provide insights on how the model makes correct responses.

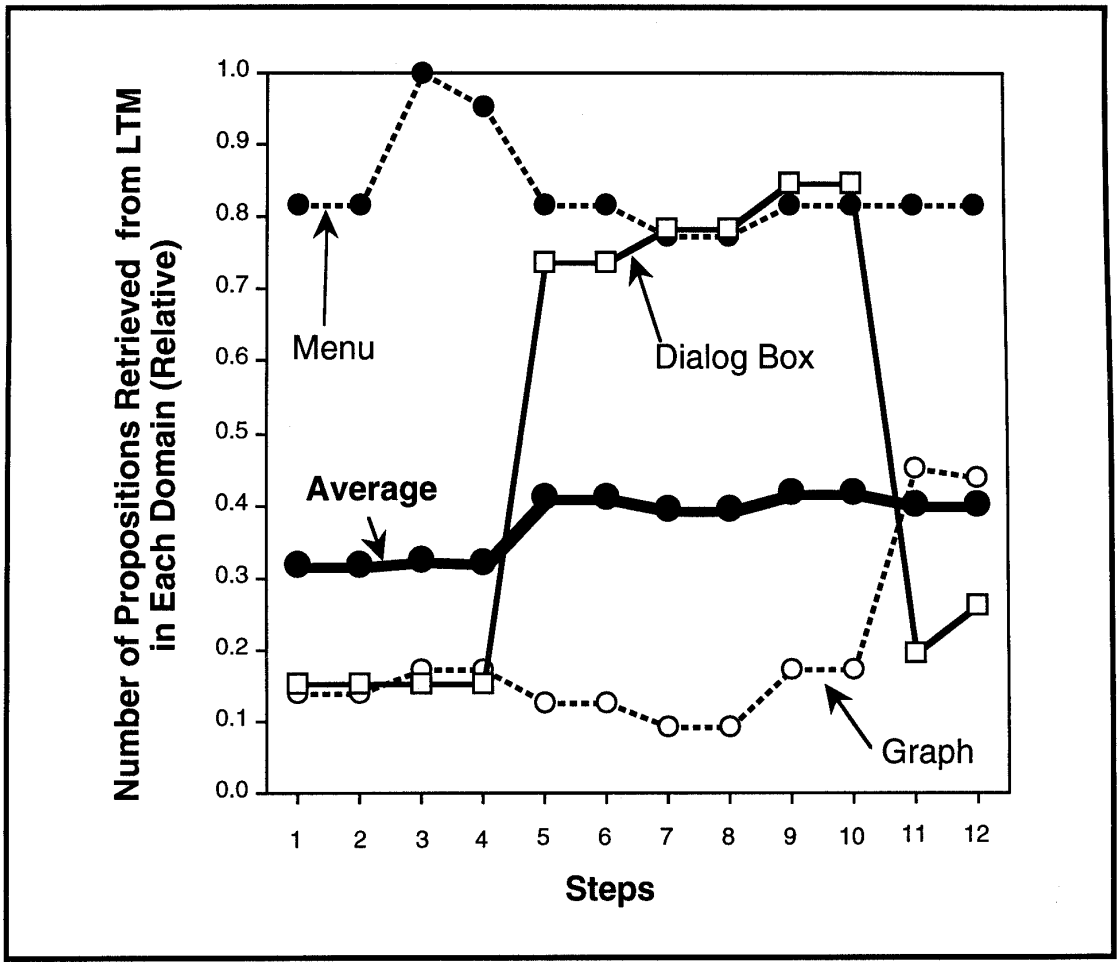


Figure 8. Results of Experiment 3. This figure plots those domains where the number of retrieved proposition varied depending on the contents of the display and the current goals.

4.3.3 Experiment 4 - Errors and Recovery from Errors

We ran 10 simulations of the Cricket Graph Task for each of the following values of the elaboration parameter : 4, 8, 12, 16, 20, and 24. The remaining values of the parameters were set equal to those used in Experiment 3. The action generated by the model at each step was recorded, and then the model was provided with the displays that the selected actions generated for Steps 2, 6, 8, and 10 or the correct displays for Steps 1, 3, 4, 5, 7, 9, 11, and 12, and correct task and device goals.

4.3.3.1 Results

Figure 9 plots the probabilities of success at each step for large values of the elaboration parameter (16, 20, and 24, the solid squares connected by a dotted line), and for small values of the elaboration parameter (4, 8, and 12, the open circles connected by a solid line). Note that the probability of an error varies widely, especially for small values of the elaboration parameter. Furthermore, recall that the model is sampling from different domains of knowledge in long-term memory as a function of what is on the display at each step (See Figure 8).

4.3.3.2 Detailed Analysis of Errors

Examination of Figure 9 shows that there are large variations in error rate especially for small values of the elaboration parameter. The model found Step 3, releasing on the menu item **Line-Graph**, steps involving interactions with the dialog box, and Steps 11 and 12, the actions involved in editing the **Graph-Title**, to be particularly difficult. Selection of the correct action for these steps was not determined by

the task and device goals and the display. The network had to be elaborated by information in relevant domains contained in long-term memory. In this section, we describe how the model makes errors, and in the process, we will gain a better understanding of how the model simulates correct actions.

There are three possible causes of errors in generating an action for the next step. The first is that the process of selecting candidate objects for action during the first of two construction-integration cycles fails to include the correct object on the list of candidate objects. This process was not a major cause of errors because it is primarily controlled by the very strong links between the correct task and device goals and the unelaborated display representation.

The second is that the elaboration process fails to incorporate all of the conditions for the correct action in the elaborated display representation. The elaboration process is the major source of errors.

The third possible cause of errors is that the correct action fails to become the highest activated action among executable cognitive actions. The second and third sources are highly confounded because they are both strongly influenced by the elaboration process.

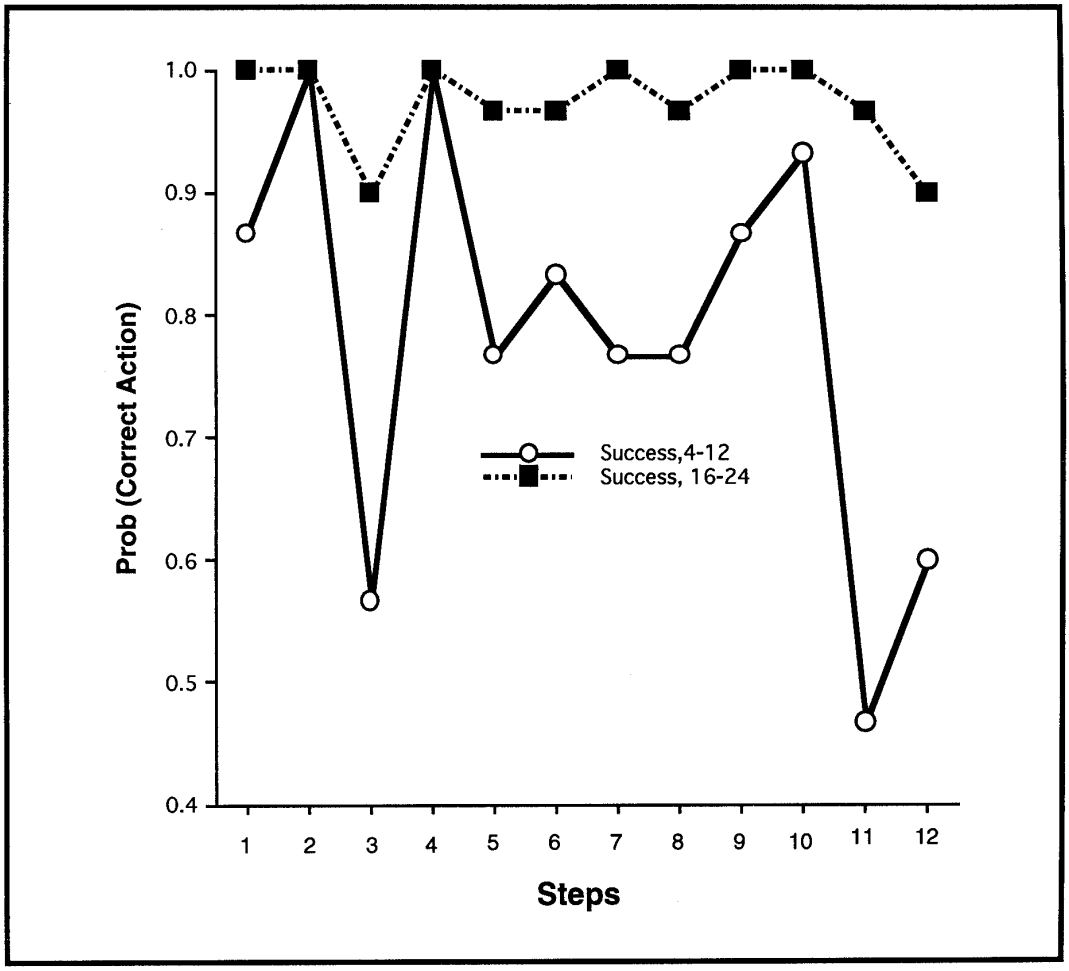


Figure 9. Probability of an error plotted as a function of Step for large and small values of N_{sample}.

4.3.3.3 Most Frequent Kind of Errors

The structure of the correct action sequence is critical to understand Figure 9. On odd numbered steps, the model must move the cursor to point at a new display object. Even numbered steps involve taking an action on the pointed-at object. Note that we scored the model as being correct even if it had not selected the correct object on odd numbered steps. Study of Figure 9 shows that the model is most likely to make errors when moving the cursor to a new display object.

4.3.3.4 Errors at Different Steps

On step 1, the correct action is to move the cursor to **Graph** in the menu bar. The competing cognitive actions on a simulation run are to move the cursor to the highlighted cell in the spreadsheet shown in Figure 3 or to **Edit** in the menu bar. The model is able to select the correct action because the cognitive action with the target of **Graph** is the most highly activated action because of its links to the current task and device goals (see Table 2).

The model makes an error on this initial menu selection when it does not correctly elaborate the display object for **Graph.** The correct action is impossible because the elaboration process did not incorporate the conditions necessary for **Graph** to be the target of the move cognitive action. The most frequent error is to select **Edit** from the menu bar.

Figure 9 shows that the model always performed the correct action on Step 2. The knowledge that must be retrieved from long-term memory in order to select **Edit** from the menu bar, the most common wrong response on Step 1, will enable the model to perform Step 2, *Press and Hold*, correctly.

On Step 3, the correct action is to drag the pointer to the menu item **Line-Graph**. Here again the model had difficulty in performing the correct action for small values of the elaboration parameter. The correct action for Step 3 requires three propositions to be retrieved from long-term memory to elaborated **Line-Graph**. The probability of simultaneously sampling of the necessary propositions is low for small values of the elaboration parameter. The error is caused by failing to satisfy the conditions for the cognitive action that points at an item in a pull-down menu.

Very similar patterns occurred at Steps 5, 7, and 9, which are all involved in interacting with the dialog box for variables selection. The correct objects to be pointed at on Steps 5, 7, and 9 are **Serial-Position(X), Observed(Y)** and **New-Plot**, respectively. The high error rates shown in Figure 9 are caused by the sampling process failing to properly elaborate the representations of the correct objects. On those simulation runs another object in the dialog box was pointed at, this incorrect action was always followed by the correct action on the next step, *Single-Click*.

Steps 11 and 12, pointing at **Graph-Title** and double clicking to edit it, are particularly difficult for the model. This situation illustrates two sources of difficulty in selection of the correct action. First, there is no display object representing the correct action; the correct action is hidden. Two propositions from long-term memory must be included in the elaborated display representation to link the display object representing **Graph-Title** with the correct action (*Double Click*) and to satisfy the conditions of the *Double Click* cognitive action .

Second, there is a very powerful competitor for the model's attention. The task and device goals overlap with both **Edit** and **Graph-Title**. Because of the argument overlap, there are very strong links with propositions in long-term memory associated with both **Graph-Title** and **Edit**. The model attempts to elaborate the display representation of both **Edit** and **Graph-Title**. The combined effect of the correct action being hidden and an incorrect action being strongly associated with the goal causes the model to make errors for low values of the elaboration parameter on both Steps 11 and 12.

4.3.3.5 Recovery From Errors

In the preceding paragraph, we showed why the model is likely to make errors on Steps 11 and 12 for low values of the elaboration parameter. If the model makes a failure, it selects **Edit** on 15 out 16 errors made in 30 simulation runs. When the model selects the wrong action pointing at **Edit** in Step 11, it always followed with *Press and Hold* as the next action. This is the same result we saw in Step 2 when the model would always select *Press and Hold* after selecting a wrong item on the menu bar.

We did a simulation experiment for these last two steps showing that the model can recover from these errors. When the pull-down menu for **Edit** appeared, the model goes through an action selection process involving two complete construction-integration cycles. In the first cycle, it selects a new list of candidate objects using the elaborated display

representation, and then it attempts to select a correct action on one of the three candidate objects. When the model considers the items on the **Edit** menu, none of them overlaps with task or device goals. Thus, the model does not include any of the **Edit** menu items on its list of candidate objects. However, it does include **Graph-Title** on the list.

The model then moves the mouse cursor to **Graph-Title** and correctly selects the next action, *Double Click*. The selection of candidate objects during the first construction-integration cycle enables error recovery. If the model selects a wrong menu, none of the items on that menu will overlap with task and device goals. Thus, none of the items on the incorrect menu will be candidates for action. The model will consider other objects on the screen as candidates for action which enables it to recover from the error.

What is particularly intriguing about this last simulation result is that we did not incorporate into the architecture any special mechanisms for error recovery. Error recovery is a consequence of the regular action planning mechanisms of the model. Also note that if the incorrect menu selection exposed a menu item that overlapped with task and/or device goals, the model would continue on this erroneous course of action .

4.4 Summary

The original goal of our simulation experiments was to demonstrate the sufficiency of the model, that is, to show that it can generate correct action sequences. Then, we examined the details of its performance to get additional insights into display-based human-computer interaction. Our second set of results, in part serendipitous, was the discovery of mechanisms by which the model accounts for errors in expert performance (rates in the range from 5% to 20%) and that it can explain some forms of error recovery. First, we deal with the results demonstrating sufficiency.

4.4.1 The Processes of Display-Based Human-Computer Interaction

The first and third experiment showed that the model is capable of generating the correct action sequences even for a realistically complex task like the Cricket Graph Task. We had to make several extensions to the original Mannes and Kintsch (1991) action-planning framework in order to simulate these more complex tasks.

4.4.1.1 Selection of Candidate Objects

Action selection on trials in which there has been a significant change in the display, e.g., the dropping of a pull-down menu, requires two complete construction-integration cycles. The first selects three candidate objects for action; the second cycle selects the action to be executed. In earlier versions of the model (Mannes and Kintsch, 1991), the simulation was simply provided with a list of three candidate objects at each step. However, these experiments used very simplified representations of the external world. There were only a few candidate objects, and selecting three out of four or five was not a distortion of the processes being simulated.

The environment we simulated, especially the Cricket Graph Task, was more complex. Any principled model of display-based human-computer interaction involving situations where there was a large, cluttered screen with many candidates for possible action would have to include mechanisms that select a subset as possible candidates for action. We found that the construction-integration process can successfully select candidate objects.

4.4.1.2 The Goal Magnification Factor

In order to get the model to successfully perform any task, we had to introduce a new parameter that multiplied the strengths of the links between the goals and the rest of the propositions in the network. These link strengths are a factor of 16 stronger than the remaining links interconnecting arguments in the network. The disparity is necessary for the model to be able to perform correctly.

These very large link strengths impact two processes in the model. First, the elaboration process, incorporating propositions sampled from long-term memory into the network, is determined by link strengths. Thus, propositions in long-term memory sharing arguments with the task and device goal representations are preferentially sampled for inclusion in the representation that determines the list of candidate objects, and, on the second construction-integration cycle, the action to be executed. Second, these very powerful link strengths mean that the task and device goals have a very strong influence on the integration process; on the candidate objects that are selected during the first construction-integration cycle, and on the highest activated action that is actually executed during the second construction-integration cycle.

4.4.1.3 The Role of the Display

The third experiment also gave us additional insights into how the screen representation guided generation of the correct action sequence. We found (see Figure 8) that the proportion of propositions in each of the

separate domains retrieved from long-term memory changed quite dramatically depending upon both the contents of the screen and the current task and device goals.

The model would incorporate into the network knowledge from long-term memory directly related to objects on the screen. For example, the model preferentially samples propositions about dialogue boxes if a dialogue box was present on the screen. This behavior, combined with the model's ability to differentially sample propositions relevant to arguments in the device and task goals, meant that propositions that were related both to a display object, like a dialogue box, and to the device and task goals would be very likely to be retrieved from long-term memory and included in the network.

The knowledge retrieved from long-term memory formed links between display objects, device and task goals, and actions. It was very often the case that the correct action had multiple links between sources of activation, i.e., task and device goals, and screen objects, and thus the correct action would win out in the competition for activation with other actions.

4.4.2 Errors

The fact that the display elaboration process could generate errors was a surprise discovery in these simulation experiments. We had expected that inappropriate choices of parameters would cause the model to fail. The results from Experiment 1 shown in Figure 5 confirmed our expectations. We had not predicted that the elaboration process would generate errors.

Previous applications of the construction-integration model to text comprehension (Kintsch, 1988) and action planning (Mannes and Kintsch, 1991; Doane, et al., 1992a; Doane, et al., 1992b) had not discovered any impact of the memory sampling process. The memory sampling process was originally incorporated into the model to account for the fact that only part of our extensive knowledge about concepts in a paragraph will be actually brought to bear during any cycle during the comprehension process. It turns out, especially for text comprehension, that the resulting representation is redundant and therefore quite robust.

Representations used during action planning for our tasks are very brittle. The model makes errors if the elaboration parameter is not set to a large value, resulting in incomplete incorporation into the network of all relevant knowledge from long-term memory.

4.4.2.1 Speed-Accuracy Tradeoff

As we stated in Section 1.3, a puzzling and frequently ignored fact in human-computer interaction literature is that experts have surprisingly high error rates, in the range of 5% to 20%. Card, et al. (1983) proposed that experts accept high error rates in order to increase their productivity, because for them error recovery can be done easily and rapidly. They trade speed for accuracy, causing slips.

We argue that the elaboration parameter describes a speed-accuracy tradeoff process where low values of the parameter reduce the amount of time taken by the elaboration process. Low values of the elaboration parameter cause the model to fail to incorporate all the conditions for the correct action in the elaborated display representation. This is the major source of errors. Parameter values in the range of 4 to 12 cause the model to simulate error rates in the range of 10% to 20%. Our model claims that many experts' errors are slips caused by failure to generate complete representations of objects on the screen.

4.4.2.2 Attention Failures

Sellen (1990) reviews classes of models that provide principled, qualitative accounts for errors. She argues that all of these models have a hierarchical representation of action sequences that include representations of top-level task goals and lower-level goals that actually control execution of elementary actions.

Reason (1990) and many others argue that control of attention is a critical determinant for generating correct performance from such a hierarchical representation. Failure to adequately attend to the ongoing process and coordinate the interaction between the various schema causes a wrong low-level schema to become activated, generating related but incorrect actions for the current task. In human-computer inetraction tasks, the users could be focusing on the task of composing new text or drawing a figure, and so on. This would lead to insufficient attention being allocated to subtasks involved in operating the interface.

Small values of the goal magnification factor cause the model to make errors because it does not focus on information related to user's goals during the elaboration process. Thus the model can simulate failures of attention.

5. CONCLUSIONS

This paper describes three sets of theoretical results.

The first is a synthesis of a diverse collection of writings on display-based human-computer interaction and problem solving including Hutchins, et al. (1986), Larkin (1989), Shneiderman (1982), and papers describing the development of the Xerox STAR (Smith, et al, 1982; Bewley, et al, 1983). The key assumption incorporated in this paper is that skilled action involves an action cycle consistent with the framework presented by Norman (1986, 1988) and incorporated into the Hutchins, et al. analysis of display-based human-computer interaction.

The second set of theoretical results is the development of a model within a cognitive architecture defined by Kintsch's (1988) construction-integration model of text comprehension. We made two important extensions to the Mannes and Kintsch (1991) model of action planning. First, our model simulates an environment in which there is a complex display with many irrelevant objects. Our model selects a small number of objects for possible action and focuses its attention on relevant information in the display and knowledge stored in long-term memory. Second, we modified the Mannes and Kintsch action representation. We assumed a richer structure on actions distinguishing between physical actions and cognitive actions. Another important difference is that we defined actions at a much smaller grain size: cursor movements, single clicking and double clicking the mouse button, and the like.

The resulting model of skilled performance is strikingly different from typical models of expert performance and error (Anderson, 1993; Reason, 1990; Card, et al., 1983). Typical models assume that skilled performance is mediated by detailed, large grain size action plans stored in long-term memory. Card, et al. (1983) refers to them as methods; Reason (1990) assumes that skilled performance is mediated by action schemata (Norman, 1981). Bovair, et al. (1990) showed that a rule-based model can define methods or action schemata as tightly integrated collections of 5 to 15 rules.

Even more radical is the assumption that skilled performance involves the rapid generation of correct action sequences and not retrieval of stored rules or schemata from long-term memory. Action planning is assumed to be analogous to text comprehension (Mannes and Kintsch, 1991) and is controlled by knowledge retrieved from long-term memory that links display objects, goals, and the correct action. Correct actions that are hidden, or whose labels are inconsistent with the representation of goals are linked by additional knowledge that must be retrieved from long-term memory.

Our third and most important set of theoretical results was to show that the model provides a principled explanation of errors made by expert users. The model is not consistent with a large fraction of current theories of skilled performance. However, it provides a principled explanation for the fact that error rates in skilled human-computer interaction are the range of from 10% to 20%. In addition, it also accounts for the fact that errors are not random; the incorrect action is often related to the correct action (Reason, 1990).

When the model makes an error, it has attempted to select a correct action based on incomplete knowledge. The incorrect action will be highly constrained by the user's current goals, the current state of the display, and the partial knowledge that was successfully retrieved from long-term memory. The model is also capable of recovery from errors. What is important about all of these results is that they are consequences of the basic architecture of the model, i.e., skilled use involves the rapid generation of the correct action sequence and that actions are defined at a small grain size.

There are additional implications of the model. In a related research program, Polson, Lewis, Rieman, and Wharton (1992) have used the construction-integration framework to propose a model of learning by exploration. The Polson, et. al. model assumes that correct actions with labels that are unrelated to the user's goals or that are hidden will be difficult to acquire by exploration and might be difficult to remember. The model of skilled performance presented in this paper can successfully perform such actions. However, the knowledge required to link the display objects, goals, and poorly labeled or hidden actions must be retrieved from long-term memory.

For the novice, this linking knowledge is difficult to discover by exploration and thus will be hard to acquire and may be difficult to retain. For the expert, performance of an action that involves such links requires their successful retrieval from long-term memory. These simulations show that such steps are error prone. Thus, there is a direct link between actions that are hard for novices to learn and for experts to perform reliably.

In summary, we synthesized a variety of views on display-based, human-computer interaction into a

single model using the construction-integration architecture. The architecture has provided us with a principled explanation for the high error rates of expert users. In addition, the architecture also enables us to give a principled account of the fact that errors are not random but are closely related to the appropriate action. Finally, combining our results with related work on learning by exploration leads to the intriguing claim that steps that are difficult to learn will tend to be highly error prone.

ACKNOWLEDGMENT

The authors gratefully acknowledge research support from the National Science Foundation, grants numbers IRI 87-22792 and IRI 91-16640, and by Army Research Institute Contract MDA903-89-K-0025. The opinions expressed in this paper are those of the authors and not necessarily those of any of the supporting organizations. Muneo Kitajima was a visitor at the Institute of Cognitive Science, University of Colorado during our collaboration. His participation was supported by New Energy and Industrial Technology Development Organization. We thank John Rieman and Clayton Lewis for ideas, direction, and comments on earlier versions of this paper. Walter Kintsch, Richard Young, and Andrew Howes also made important contributions to our thinking.

REFERENCES

ANDERSON, J. R. (1993). *Rules of the mind.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

BEWLEY, W. L., ROBERTS, T. L., SCHROIT, D., & VERPLANK, W. L. (1983). *Human Factors Testing in the Design of Xerox's 8010 'Star' Office Workstation: Case Study D: The Star, the Lisa, and the Macintosh.*

BOVAIR, S. & KIERAS, D. E. (1985). A guide to propositional analysis for research on technical prose. In BRITTON, B. K. & BLACK, J. B., Eds. *Understanding expository text.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

BOVAIR, A. S., KIERAS, D. E. & POLSON, P. G. (1990). The acquisition and performance of text-editing skill: a cognitive complexity analysis. *Human-Computer Interaction,* **5**, 1, 1-48.

CARD, S. K., MORAN, T. P., & NEWELL, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

CHAPMAN, D. (1987). Planning for Conjunctive Goals. *Artificial Intelligence,* **32**, 333-377.

DOANE, S. M. , MANNES, S.M. , KINTSCH, W. , and POLSON, P.G. (1992a) Modeling User Action Planning: A Comprehension Based Approach. *User Modeling and User-Adapted Interaction,* **2**, pp. 249-285.

DOANE, S. M., MCNAMARA, D. S., KINTSCH, W., POLSON, P. G., & CLAWSON, D. M. (1992b). Prompt comprehension in UNIX command production. *Memory and Cognition,* **20, 4**, 327-343.

HANSON, S. J., KRAUT, R. E., & FARBER, J. M. (1984). Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems,* **2**, 42-57.

HOWES, A. (1993). Recognition-based problem solving. In *Proceedings of the 15th Annual Meeting of the Cognitive Science Society.* Boulder, Colorado. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

HOWES, A. & PAYNE, S. J. (1990). Display-based competence: towards user models for menu-driven interfaces. *International Journal of Man-Machine Studies,* **33**, 637-655.

HOWES, A. & YOUNG, R. M. (1991). Predicting the learnability of task-action mappings. In *Proceedings of the CHI'91 conference on human factors in computing systems.*, 113-118. New York: ACM.

HUTCHINS, E. L., HOLLAN, J. D., & NORMAN, D. A. (1986). Direct manipulation interfaces. In NORMAN, D. A. & DRAPER, S. W., Eds. *User Centered System Design,* 87-124. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

JOHN, B. E. & VERA, A. H. (1992). A GOMS analysis of a graphic, machine-paced, highly interactive task. In *Proceedings of the CHI'92 conference on human factors in computing systems,* 251-258. New York: ACM.

KIERAS, D. E. (1988). Towards a Practical GOMS Model Methodology for User Interface Design. In M. Helander (Ed.) *The Handbook of Human-Computer Interaction.* Amsterdam, NV: North-Holland.

KINTSCH, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review,* **95**, 163-182.

KITAJIMA, M. & POLSON, P. G. (1992). A computational model of skilled use of graphical user interfaces. In *Proceedings of the CHI'92 conference on human factors in computing systems.*, 241-249. New York: ACM.

LARKIN, J. H. (1989). Display-based problem solving. In KLAHR, D. & KOTOVSKY, K., Eds. *Complex Information Processing: The Impact of Herbert A. Simon.*, 319-342. Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

LARKIN, J. H. & SIMON, H. A. (1987). Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science*, **11**, 65-100.

MANNES, S. M. & KINTSCH, W. (1991). Routine computing tasks: Planning as understanding. *Cognitive Science*, **15**, 305-342.

MANNES, S. & ROUSHEY, M. (1990). NETWORK: A computer simulation of the construction-integration model. *ICS Technical Report* #90-13. Boulder CO: Institute of Cognitive Science, University of Colorado.

MAYES, J. T., DRAPER, S. W., MCGREGOR, M. A., & OATLEY, K. (1988). Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens. In JONES, D. M. & EINDER, R., Eds. *People and Computer IV.* Cambridge, UK: Cambridge University Press.

NEWELL, A. & SIMON, H. (1972). *Human Problem Solving*. Englewood Cliffs, New Jersey: Prentice-Hall.

NORMAN, D. A. (1981). Categorization of action slips. *Psychological Review*, **88**, 1-15.

NORMAN, D. A. (1986). Cognitive Engineering. In NORMAN, D. A. & DRAPER, S. W., Eds. *User Centered System Design*, 31-61. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

NORMAN, D. A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.

PAYNE, S. J. (1991). Display-based action at the user interface. *International Journal of Man-Machine Studies*, **35**, 275-289.

PAYNE, S. J., SQUIBB, H. R., & HOWES, A. (1990). The nature of device models: The yoked state hypothesis and some experiments with text editors. *Human-Computer Interaction*, **5**, 4, 415-444.

PECK, A. V. & JOHN, B. E. (1992). Browser-SOAR: A computational model of a highly interactive task. In *Proceedings of the CHI'92 conference on human factors in computing systems.*, 165-172. New York: ACM.

POLSON, P.G., LEWIS, C., RIEMAN, J., and WHARTON, C. (1992). Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *International Journal of Man-Machine Studies*, **36**, 741-773.

RAAIJMAKER, J. G. & SHIFFRIN, R. M. (1981). Search of associative memory. *Psychological Review*, **88**, 93-134.

REASON, J. (1990). *Human Error.* New York, NY: Cambridge University Press.

SHNEIDERMAN, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behavior and Information Technology*, **1**, 237-256.

SELLEN, A. (1990). Mechanism of human error and human error detection. *Unpublished doctoral dissertation.*

SMITH, D. C., IRBY, C., KIMBALL, R., VERPLANK, W. L., & HARSLEM, E. (1982). *Designing the Star User Interface: Case Study D: The Star, the Lisa, and the Macintosh.*

SUCHMAN, L. A. (1987). *Plans and situated action: The problems of human machine communication.* New York: Cambridge University Press.

APPENDIX A

A.1. Construction Process

At the first construction process, in the object selection process, the following formula defines the strengths of links for P_i, $P_j \in$ {task goals, device goals, display representation, retrieved information from long-term memory, candidate objects};

For $i \neq j$;

$$W_{P_i,P_j} = W_{P_j,P_i} =$$

$$\left(W_{arg} \times (\text{number of shared arguments}) + \begin{cases} W_{assoc}; & \text{if } P_i \text{ and } P_j \text{ are associates} \\ 0.0; & \text{otherwise} \end{cases} \right) \times$$

$$\begin{cases} F_{task}; & \text{if } P_i \text{ or } P_j \in \{\text{task goals}\} \\ 1.0; & \text{otherwise} \end{cases} \times$$

$$\begin{cases} F_{device}; & \text{if } P_i \text{ or } P_j \in \{\text{device goals}\} \\ 1.0; & \text{otherwise} \end{cases}$$

For $i=j$;

$$W_{P_i,P_j} = 1.0$$

At the second construction process, in the action selection process, the above holds for P_i, $P_j \in$ {task goals, device goals, display representation, retrieved information from long-term memory}. In addition, for the same set of P_i and, $L_k \in$ {specific cognitive actions},

$$W_{P_i,L_k} = W_{L_k,P_i} =$$

$$(W_{arg} \times (\text{number of shared arguments})) \times \begin{cases} F_{task}; & \text{if } P_i \in \{\text{task goals}\} \\ 1.0; & \text{otherwise} \end{cases}$$

$$\times \begin{cases} F_{device}; & \text{if } P_i \in \{\text{device goals}\} \\ 1.0; & \text{otherwise} \end{cases}$$

In case that all consequences of L_k have already been found in the elaborated display representation, links for $P_i \in$ {consequences of L_k} are replaced by

$$W_{P_i,L_k} = W_{\text{action-i}}$$

And, asymmetric causal relations are considered for L_k, $L_l \in$ {specific cognitive actions};

For $k \neq l$,

$$W_{L_k,L_l} =$$

$$\begin{cases} W_{\text{action-e}} & \text{if } L_k \text{ supports } L_l \\ W_{\text{action-i}} & \text{if } L_k \text{ inhibits } L_l \\ 0.0 & \text{otherwise} \end{cases}$$

For $k=l$,

$$W_{L_k,L_l} = 1.0$$

A.2. Integration Process

The integration process is formally defined as follows. A set of nodes that are interconnected by the construction process is represented by

$$\{S_1,\ldots,S_i,\ldots,S_N,R_1,\ldots,R_j,\ldots,R_{N'}\},$$

where S_i's are activation nodes representing task goals, device goals, and display, and R_j's are receptor nodes representing information retrieved from long-term memory, and candidate objects in the object selection process, or specific cognitive actions in the action selection process. N and N' are the number of source nodes and the number of receptor nodes, respectively.

With this indexing system, the pattern of activation after v-th flash can be expressed by a vector, $\vec{\mathbf{A}}^{(v)}$, and

the strengths in the network, by a matrix, **W**, as follows, respectively;

$$\overrightarrow{\mathbf{A}^{(v)}} \equiv (\overrightarrow{\mathbf{A}_S^{(v)}}, \overrightarrow{\mathbf{A}_R^{(v)}})$$
$$\overrightarrow{\mathbf{A}_S^{(v)}} \equiv (A_{S1}^{(v)}, \ldots, A_{SN}^{(v)})$$
$$\overrightarrow{\mathbf{A}_R^{(v)}} \equiv (A_{R1}^{(v)}, \ldots, A_{RN'}^{(v)})$$

$$\mathbf{W} \equiv \begin{bmatrix} W_{S1,S1} & \cdots & W_{S1,SN} & W_{S1,R1} & \cdots & W_{S1,RN'} \\ \cdots & & \cdots & \cdots & & \cdots \\ W_{SN,S1} & \cdots & W_{SN,SN} & W_{SN,R1} & \cdots & W_{SN,RN'} \\ W_{R1,S1} & \cdots & W_{R1,SN} & W_{R1,R1} & \cdots & W_{R1,RN'} \\ \cdots & & \cdots & \cdots & & \cdots \\ W_{RN',S1} & \cdots & W_{RN',SN} & W_{RN',R1} & \cdots & W_{RN',RN'} \end{bmatrix}$$

Where, $\overrightarrow{\mathbf{A}_S^{(v)}}$ and $\overrightarrow{\mathbf{A}_R^{(v)}}$ are vectors representing activation values of the source nodes and receptor nodes, respectively.

The initial activation values for the source nodes, $A_{S_i}^{(0)}$ (for $1 \le i \le N$), are set to 1.0, and for the receptor nodes, $A_{R_i}^{(0)}$ (for $1 \le i \le N'$), to 0.0. The activation vector after v-th activation flash, $\overrightarrow{\mathbf{A}^{(v)}}$, is defined as follows;

For activation nodes, they are reset to the constant value.

$$A_{S_i}^{(v)} = 1.0$$

For receptor nodes,

$$A_{R_i}^{(v)} = N \times \frac{\max(0.0, \widetilde{A_{R_i}^{(v)}})}{\sum_{k=1}^{N'} \max(0.0, \widetilde{A_{R_k}^{(v)}})}$$

where, unnormalized activation value, directly calculated by matrix multiplication,

$$\widetilde{A_{R_i}^{(v)}} = \sum_{j=1}^{N} A_{S_j}^{(v-1)} \times W_{S_j,R_i} + \sum_{j=1}^{N'} A_{R_j}^{(v-1)} \times W_{R_j,R_i}$$
$$= \sum_{j=1}^{N} W_{S_j,R_i} + \sum_{j=1}^{N'} A_{R_j}^{(v-1)} \times W_{R_j,R_i}$$

is normalized so as to the sum of activation stored in the receptor nodes is equal to the sum of activation in the activation nodes, N.

In order to estimate the degree of convergence of the pattern of activation, average change of the activation vector is used;

$$\varepsilon^{(v)} = \frac{1}{N'} \times \sum_{i=1}^{N'} \left| A_i^{(v)} - A_i^{(v-1)} \right|$$

When this value reaches below a criterion value, say, 0.01, the network is considered to be stabilized.

The value of normalization factor might be defined arbitrarily. However, the above equations would suggest that effectively it defines relative contribution of the source and receptor nodes in updating the activation value of receptor nodes. The smaller the normalization factor becomes, the less significant the activation value of receptor nodes becomes. The current normalization procedure would correspond to a situation where source nodes and receptor nodes equally contribute to activation of receptor nodes. There would be situations where normalization procedure should be differently prescribed.