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

Muneo Kitajima
Human Informatics Department
National Institute of Bioscience and Human-Technology
1-1 Higashi Tsukuba Ibaraki 305, Japan
Phone: +81 298 54 6731
Fax: +81 298 54 6752
e-mail: kitajima@nibh.go.jp

Peter G. Polson
Institute of Cognitive Science
University of Colorado,
Boulder, Colorado 80309-0345
Phone: +1 (303) 492-5622
e-mail: ppolson@psych.colorado.edu

ABSTRACT
This paper[1] describes a computational model of skilled use of an application with a graphical user interface[2]. The model provides a principled explanation of action slips, errors made by experienced users. The model is based on Hutchins, Holland, and Norman's (1986) analysis of direct manipulation and is implemented using Kintsch and Mannes's (1991) construction-integration theory of action planning. The model attends to a limited number of objects on the screen and then selects action on one of them, 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 display states, and knowledge about the interface and the application domain. The model simulates a graph drawing task. In addition, we describe how the model makes errors even when it is provided with the knowledge sufficient to generate correct actions.

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. Our model is synthetic in that it attempts to integrate the views of numerous researchers 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 a large simulation experiment that validates the sufficiency of the model for a realistically complex task and shows that it explains why 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: evaluation of the consequences of an action and planning an appropriate next action. The complexity of these evaluation and planning activities determines the difficulty of learning and performing a task.

[1] This paper is a revised version of Kitajima and Polson (1994), published from the Institute of Cognitive Science, University of Colorado, ICS Technical Report #94-02. This paper will appear in the International Journal of Human-Computer Studies.

[2] A preliminary version of this model was described in Kitajima and Polson (1992), a paper presented at CHI'92.

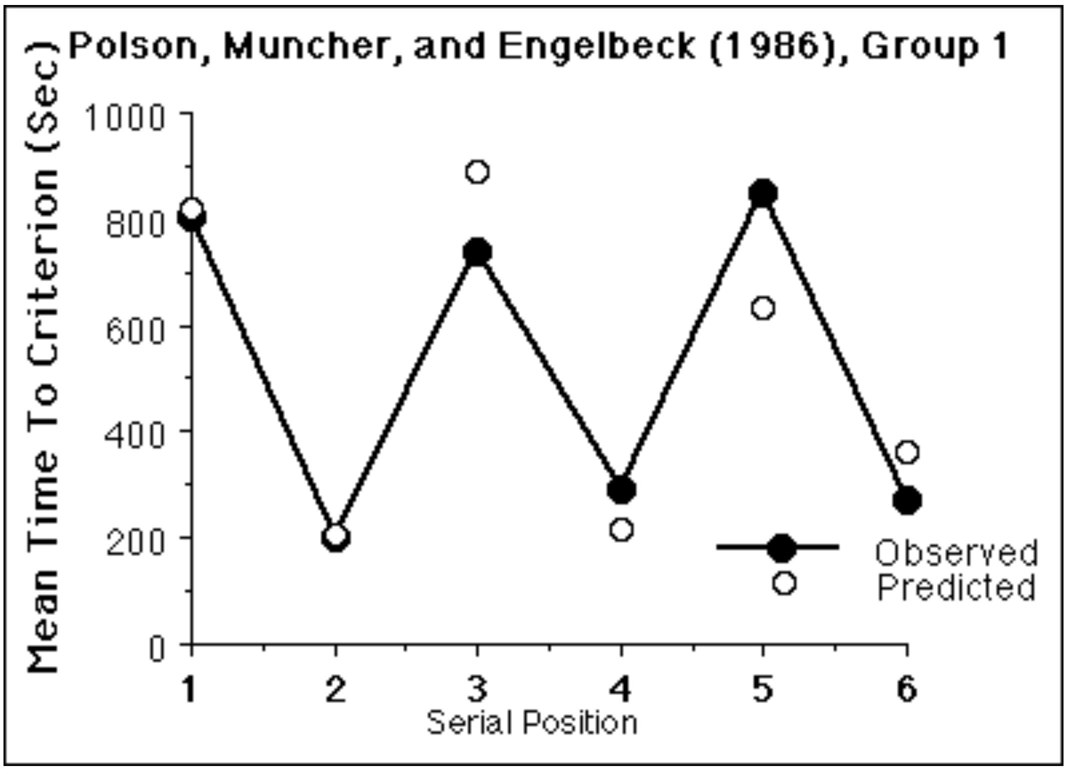


Figure 1. The graph to be produced by the user during the Cricket Graph Task.

1.2 Display-Based Problem-Solving

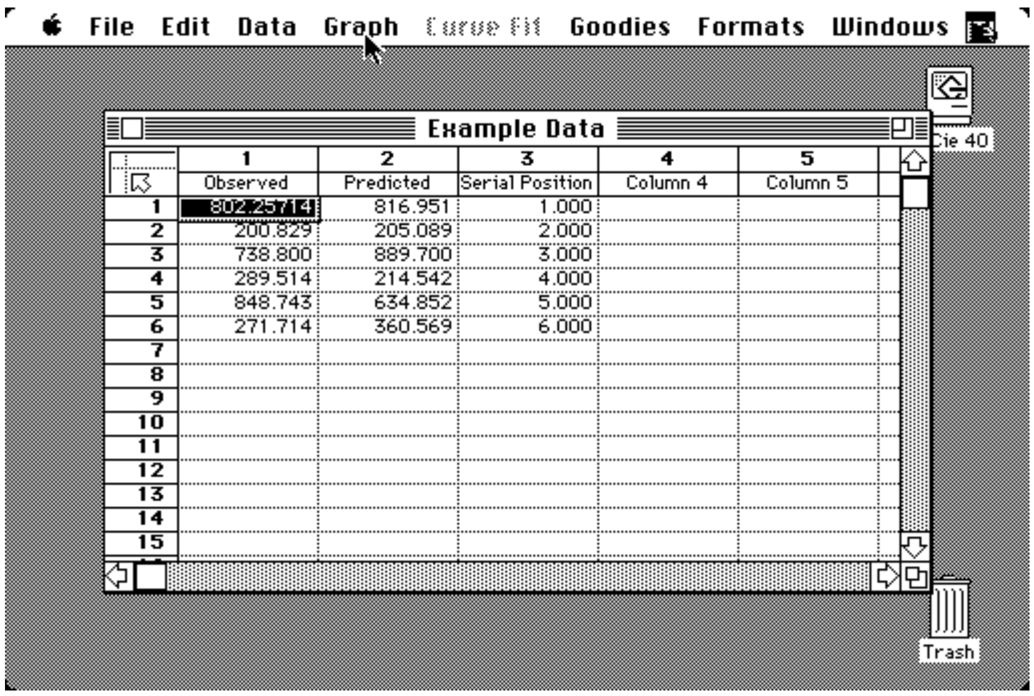
Larkin and Simon (1987) argued that displays facilitate problem-solving by allowing users to substitute perceptual operations for effortful symbolic operations, and that displays can reduce the amount of time spent searching for critical information. Larkin (1989) extended this analysis to tasks with 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 is 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 is not consistent with the process that would be required to generate 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 rejected about goal structures on the grounds that it is often difficult if not impossible to find any direct evidence for their existence. Larkin’s arguments 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.



Example Data

	1	2	3	4	5
	Observed	Predicted	Serial Position	Column 4	Column 5
1	802.25714	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 2. This is the state of the display just after the user has pointed at “Graph” menu item.

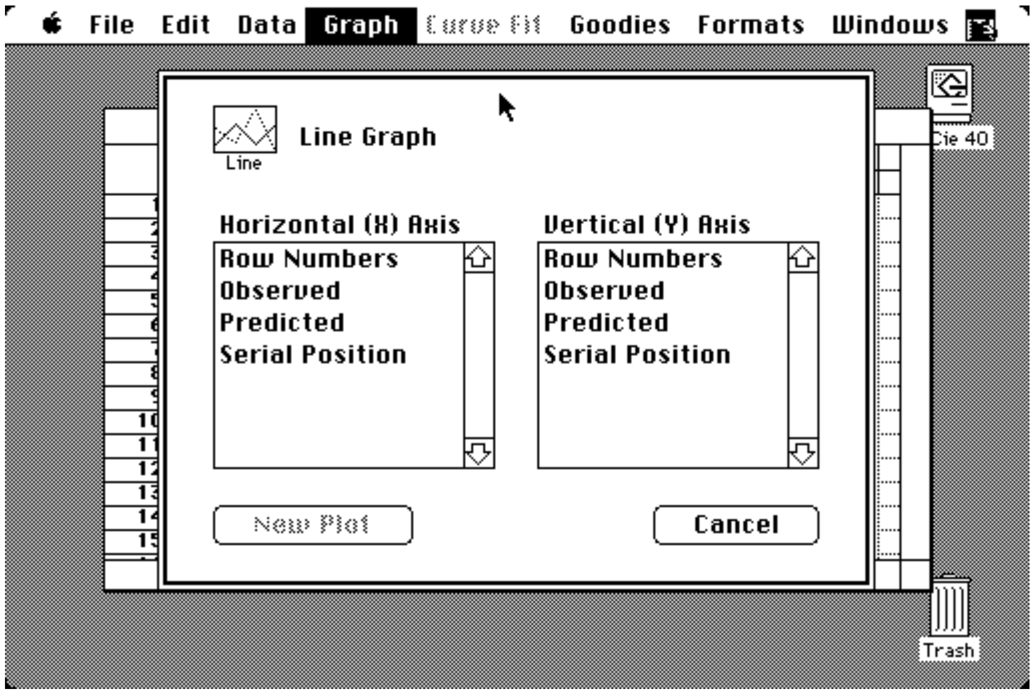


Figure 3. The critical intermediate state of the first subtask in the Cricket Graph Task. User must select “Serial Position” from the left scrolling window, “Observed” from the right, and then click “New Plot.”

1.3 Errors in Human-Computer Interaction

A puzzling and frequently ignored fact in human-computer interaction 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. Twenty-one 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. Most of these errors are action slips. 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.

1.4 An Example Graphical User Interface and An Example Task

The task used in our simulation experiment involved preparing a graph that matches an example using Cricket Graph 1.3[3]. 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 3.

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." The user's task is to plot the data and edit the resulting default graph to match the example given in Figure 1. Double-clicking "Example Data" causes the program to display a spreadsheet with three columns labeled "Observed," "Predicted," and "Serial Position." Figure 2 shows the display after the user has moved the mouse cursor to the menu item **Graph**. 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 3, 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 enable the user to transform the default graph into a graph that matches the appearance of the model.

Cricket Graph and similar programs like DeltaGraph Pro combine the functions of several different application program including spreadsheets (e.g., EXCEL) and draw programs (e.g., McDRAW). Successful use of an application like Cricket Graph requires the skills necessary to operate a significant fraction of the functionality of the Macintosh interface like creating and editing text and basic operations on spreadsheet data as well as specific knowledge about the program's interface and the task domain of statistical graphs. Thus, the results of our theoretical analyses should be generalizable to other tasks and application programs.

2. OUTLINE OF THE MODEL - A COMPLETE ACTION CYCLE

2.1 Overview of the Model

Our theory elaborates Hutchins et al. (1986) analysis of direct manipulation and Norman's (1986, 1988) action theory framework shown in Figure 4. The four basic components are: (1) goals representing what the user wants to accomplish which are a schematic outline of the sequence of subtasks 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 assumes two processes for the stage of evaluation and two for the stage of execution.

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

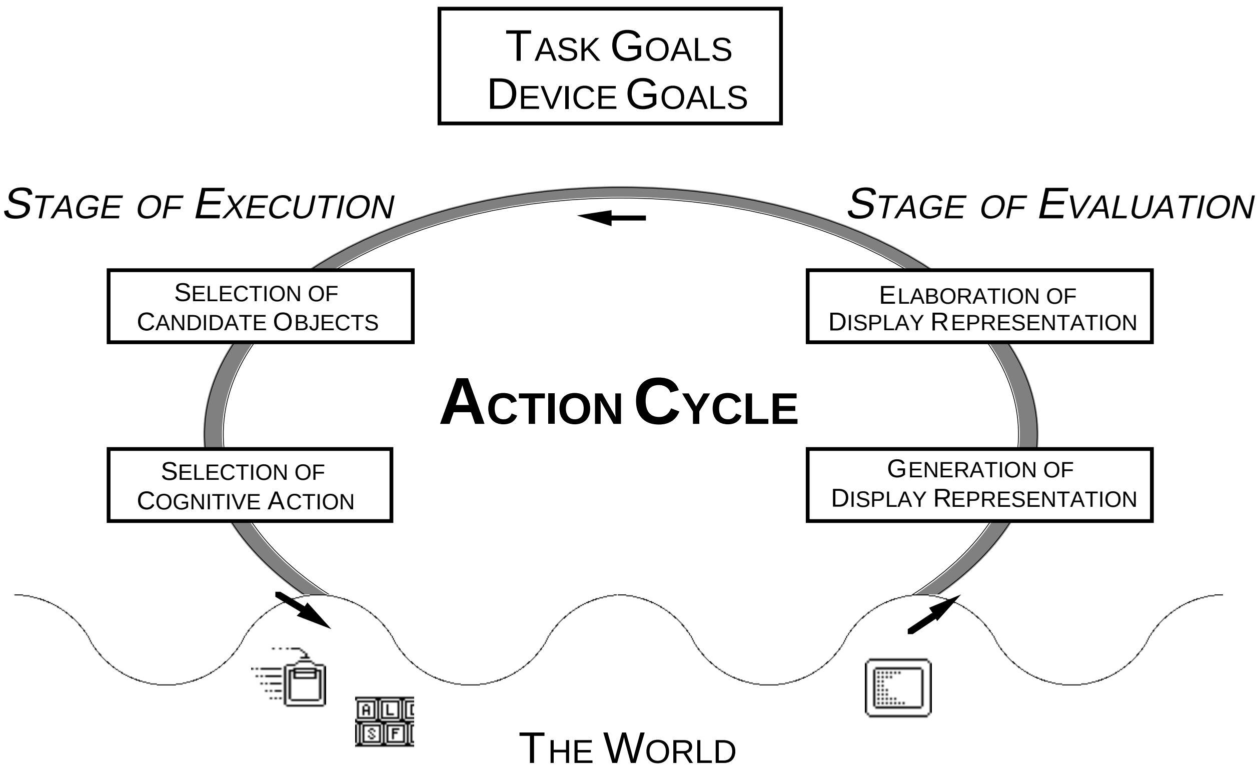

TASK GOALS
DEVICE GOALS
STAGE OF EXECUTION
STAGE OF EVALUATION
SELECTION OF CANDIDATE OBJECTS
ELABORATION OF DISPLAY REPRESENTATION
ACTION CYCLE
SELECTION OF COGNITIVE ACTION
GENERATION OF DISPLAY REPRESENTATION
THE WORLD

Figure 4. Overview of Norman's action cycle, defined by four components; goals, the stage of evaluation, the stage of execution, and the world.

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 four subprocesses shown in Figure 4. In the following sections, we briefly describe each subprocess in Figure 4.

2.2 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 3). Complete list of the task goals and device goals for the Cricket Graph Task is presented in Table 2 in Section 4.2.

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.

2.3 Stage of Evaluation

There are two processes involved in the evaluation stage. (See the right portion of Figure 2.) The first involves generation of the display representation. The second involves evaluation of information presented on the display via an elaboration process. The following sections describe these processes.

2.3.1 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.

For each object shown in the screen snapshot in Figure 2, the model creates an arbitrary identifier and then describes the object in terms of a limited number of appearance attributes. The representation of the menu item, **Graph**, for example, only includes such information as not currently being pointed at, being displayed in normal video, identifying it as a graph menu item, and so on. More details about the display representation are in Section 3.2.1.1.

2.3.2 Elaboration of the Display Representation

A key idea in the Hutchins, et al. (1986) analysis of graphical user interface is the gulf of evaluation, the difficulty of evaluating the display that results from the last action. Recall that this model's display representation contains no information about the meanings of the objects on the screen, the interrelationships between display objects, or relationships between the task and display objects. Such knowledge is critical in providing links between the current goals, objects on the screen, and the action to be performed. Building these links simulates evaluation of the display. The gulf of evaluation in our model is measured by the number of links that must be incorporated into the display representation in order to successfully select the correct action.

Building the links that bridge the gulf of evaluation is done by a *memory sampling process* that retrieves the necessary information from long-term memory using the representations of goals and the display as retrieval cues. 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 evaluation of the display that results from the last action in the context of the current task and device goals.

The evaluation process can fail even when an expert user has in long-term memory all of the information sufficient to correctly evaluate the display. This process is described in Section 3.3.1.2. The elaboration process is probabilistic. As a result, information may be omitted that is necessary to properly evaluate the display. An incompletely elaborated display representation can cause the model to make an incorrect action.

The **Graph** menu item in the initial display shown in Figure 2 can be elaborated by retrieving such knowledge as that the **Graph** menu item can be pulled down and that the **Graph** menu item has **Line-Graph** on its pull down menu. The memory sampling process can fail to retrieve either or both of these links from long-term memory. If one or both are missing, the model cannot pull down the **Graph** menu. The model would select another action, for example, pulling down **File** menu item.

2.4 Stage of Execution

The other key idea in Hutchins, et al. (1986) is the gulf of execution, the difficulty of formulating and executing the action or action sequence specified by the newly revised goals. In the model described in this paper, the stage of execution which bridges this gulf involves the two processes, selection of candidate objects and choice of one action-object pair. The selected action is performed on one object, the result of the action changes the display and/or the state of the system. These processes are shown in the left portion of Figure 2. The model represents actions at small grain size like move the mouse cursor, single click, grab and hold, and the like. Thus, in this model, the gulf of execution is small and fixed.

2.4.1 Selection of Candidate Objects

This process involves selection of three of candidate objects from the large number of display objects in the representation of the current screen. The model considers information from the goals, the display representation, and the knowledge retrieved from long-term memory in the process of selecting the three objects. This process is described in more detail in Section 3.3.2.3.

2.4.2 Selection of An Action-Object Pair

The action selection process generates all possible actions that could be carried out on the three candidate objects. It then combines information from the goals, the display representation, and the knowledge retrieved from long-term memory to select one action-object pair and updates the display representation with the consequences of the action. There are a large number of possible actions. The definition of an action combines a physical action (pointing at an object, click and double-click on pointed-at object, drag, and the like) with different versions of each of the physical action defined by different system states and intentions of the user. For example, moving the mouse cursor in order to edit a text object or moving the mouse cursor to pull down a menu are represented in the model as different actions. This process is described in more detail in Section 3.3.2.4.

In the display shown in Figure 2, the object selection process selects three menu items, **Graph**, **File**, and **Data**, as the candidate objects for the next action. The action selection process begins by generating a representation of the 149 possible combinations of the action-intentions and the three menu items. One of these possibilities is moving the mouse cursor to **Graph** with the intention of pulling down the menu.

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.

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 goals, 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

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 text 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 many alternative interpretations of a text or one out of many 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 text comprehension to action planning is to show that it demonstrates its understanding by executing actions necessary to comply with a request contained in a text.

Kitajima and Polson (1992, 1994) 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. Section 5 contains a more detailed comparison of this model with other action planning models based on the construction-integration theory (Doane, Mannes, Kintsch, & Polson, 1992a; Doane, Mcnamara, Kintsch, Polson, & Clawson, 1992b).

3.2 The Network Representation

Our model and Mannes and Kintsch (1991) represent the knowledge required to generate a correct action sequence including goals, the display, information stored in long-term memory, candidate objects, and actions as propositions. These model borrow and extend the representational machinery that have been developed for theories of text comprehension (Kintsch, 1974, 1988; Bovair and Kieras, 1985; Anderson, 1983). Such models represent meaning by interconnecting a collection of propositions into a network. In other works, these model are configural (patterns of interconnections) theories of meaning.

The model described in this paper builds two such networks during the stage of execution. The first is constructed from a collection of propositions representing the task and device goals, the display, the knowledge retrieved from long-term memory by the memory sampling process, and the candidate objects. This network is used in the process of selecting three candidate objects. The second network is constructed from the propositions representing the task and device goals, the display, the knowledge retrieved from long-term memory by the memory sampling process, and representations of all possible action-object combinations for the three candidate objects.

In the following sections, we describe the details of how the goals, display, and information retrieved from long-term memory are represented as propositions. We then show how the two networks are constructed and how the three candidate objects and an action-object pair are selected. We also present the details of the memory sampling process.

3.2.1 Display, Goals, Information in Long-Term Memory, and Candidate Objects

The display, task and device goals, information in long-term memory, and candidate objects 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 text 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)
```

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 Representations of Display

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 2. 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, (P_1)

OBJECT23 is_a_kind_of DISPLAY-OBJECT, (P_2)

OBJECT23 is_a_kind_of GRAPH-MENU-ITEM. (P_3)

and three proposition for its display status:

OBJECT23 is_pointed_at, (P_4)

OBJECT23 is_not_highlighted, (P_5)

OBJECT23 is_not_grabbed. (P_6)

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

P_1 states that the object exists and is on the screen. P_2 classifies the object as a display object. DISPLAY-OBJECT represents a class of display objects that allow for certain kinds of actions. P_3 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 P_4 to P_6 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 Representations of Goals

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 in Figure 3. The device goal is an abstract description of the display.

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 correct action sequence.

3.2.1.3 Representations of Knowledge in Long-Term Memory

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.

Possible elaborations
We already have the display representations for OBJECT23, the **Graph** menu item, that appears in Figure 2 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,

OBJECT23 is_not_a_kind_of TEXT,

OBJECT23 is_associated_with SCATTER-GRAPH-MENU-ITEM,

OBJECT23 is_associated_with LINE-GRAPH-MENU-ITEM, (P_7)

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.

3.2.1.4 Representations of Candidate Objects
The model constructs representations of candidate objects by first searching for tokens representing display objects in the elaborated display representation and then generating corresponding propositions that state that the display object is a candidate for action. In Figure 2, for example, there are ten objects that represent the menu bar. In addition, there are objects defined by elements of the window, icons, and the like. The model generates as many propositions representing candidate objects as the number of display objects represented in the display representation.

3.2.2 Representation of Actions
The model's action representation is taken from Kintsch and Mannes (1991) and combines processes from Kintsch's (1988) construction-integration theory with mechanisms from rule-based models of skill (Anderson, 1993). The representation of a specific action-object pair is generated by combining information about the object to be acted on, the function of the action, the physical constraints that must be satisfied for the action to take place, the physical action involved, and the consequences of the action.

Recall that actions are defined at a small constant grain size that is defined by the characteristics of the physical actions involved. In the current model, physical actions simulated are *Move Mouse Cursor*, *Single Click*, *Double Click*, *Press and Hold Mouse Button Down*, *Release Mouse Button*, and *Type*.

The complete specification of an action on an object is generated by combining a given physical action-object pair with the description of the different functions of that action on the object. For example, the action, *Press and Hold Mouse Button Down*, can be combined with the function of showing pull-down menu, or with the function of dragging the selected object. *Move Mouse Cursor* can be combined with the function of changing the cursor shape to the arrow, or to the I-beam. *Single Click* can be combined with the function of changing the object state to the selected or to the deselected.

The conditions necessary for a physical_action-object-function combination to be executed are defined by states of the object, such as whether or not it is pointed-at, whether or not it is text, whether or not it is highlighted, whether or not it is grabbed, etc., and by a proposition stating that the object has the function. The consequences of the execution of the physical_action-object-function combination are defined by states of display objects. Such combinations of physical_action-object-function generate six representations 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*, a total of 18.

A variablized version of each action representation is bound to each of the three candidate objects. This process occurs without any consideration of whether the resulting action-object representation can be executed in the current context. An action-object pair can be executed if the current display satisfies the conditions for its execution.

Table 1. Summary of relationships between model's processes and representations.

Representations	**PROCESSES** elaboration	candidate object selection	action-object pair selection
task and device goals, and display	used as retrieval cues	incorporated in the network / used as source of activation	incorporated in the network / used as source of activation
long-term memory	retrieved by a probabilistic memory sampling process	retrieved propositions are incorporated in the network	retrieved propositions are incorporated in the network
candidate objects	*not used*	incorporated in the network / most highly activated nodes represent candidates objects for next action	*not used*
action-object pairs	*not used*	*not used*	incorporated in the network by using the selected candidate objects / most activated eligible node represents the next action

The action-object representation has three components. The first is a proposition whose first argument is the associated physical action followed by several arguments that enumerate major features of the action-object representation in terms of the display object to be acted on and its functions. The second component is a set of conditions, like the conditions of a rule-base representation (Anderson, 1993), tested to determine executability of the action in the context defined by the display representation and information retrieved from long-term memory. The third component is a set of propositions that are added to the network when the action is executed representing the consequences of action.

The following is an example of an action-object representation for pointing at **Graph** menu.

```
Name:   Point-at Graph In Menu-Bar with ARROW-
        Shaped Cursor
Condition: IF
        Graph is on Screen
        Graph is not Grabbed
        Not Pointing at Graph
        POINTER-SHAPE is ARROW
        POINTER-SHAPE is NOT I-BEAM
        Graph is NOT TEXT Object
Action:
        Pointing at Graph
        Graph is on Screen
        Graph is not Grabbed
        POINTER-SHAPE is ARROW
        POINTER-SHAPE is NOT I-BEAM
```

3.2.3 Summary

Table 1 illustrates the relationships between various representations and the processes. In the second column, we have the elaboration process where the representations of the goals, the display and long-term memory are relevant.

3.3 The Complete Action Cycle

This section summarizes how the action cycle model is implemented using the construction-integration theory as outlined in Figure 4. The stage of evaluation involves two processes: generation of the display representation and elaboration of the display representation via the memory sampling process. These processes take place only if the last action lead to a major change in the state of the display like the appearance of a dialog box, pull-down menu, or highlighting of a selected item in a list. These processes do not occur after the movement of the mouse pointer. The stage of execution involves two processes: selection of three candidate objects and selection and execution of one action-object pair.

The typical action sequence required to perform the Cricket Graph Task simulated in our experiment can be segmented into pairs of actions. The first is a mouse cursor movement. The second is an action that leads to a major display change. The all of the processes in the evaluation and execution stages are executed when selecting a target of a mouse cursor movement. The model simulates evaluation of the last major display change caused by the second action of the previous pair and simulates execution of the movement of the mouse cursor to a new object. The change of the location of the mouse cursor only involves updating the display representation of the location of the pointer. The model then simulates selection and execution of an action on the newly pointed at object.

3.3.1 Stage of Evaluation

3.3.1.1 Build the Display Representation

The processes involved in constructing the display representation are not simulated in the model. The lists of propositions describing the screen after each major display change were constructed by hand and stored in a file along with the associated task and device goals. The content of this representation is described in Section 3.2.1.1. The list of propositions describing each screen was read by the program after each action that lead to a major display change. Thus, the model always had the correct task and device goals, and the simulation behaved as if the simulated user had always made the correct action leading to a display change.

3.3.1.2 Memory Sampling Process

Elaboration of the display that results from a major change is simulated by a memory sampling processes taken from Kintsch’s (1988) construction-integration model. Kintsch used the Raaijmaker and Shiffrin (1981) model of memory retrieval to simulate the process of retrieving information from long-term memory for incorporation into the network. Kintsch assumed that the propositional representation of a sentence momentarily activates the meanings of words and other related information in long-term memory. This knowledge is incorporated in the network during the construction phase. Kintsch and Mross (1985) present evidence in favor of these assumptions.

Recall that the representation of each display object contains no information about the function of an object, legal actions on a object, or relationships between objects. This information is stored in long-term memory in the form of propositions. The content and format of this information is described in Section 3.2.1.3. This information is used to elaborate the display representation.

The collection of proposition representing the goals, display, and content of long-term memory can be thought of as a collection of linked items. The links are defined by shared arguments in pairs of propositions. For example, all of the proposition described in Sections 3.2.1.1 to 3.2.1.3 describing `OBJECT23`, the **`Graph`** menu item, contain the argument `OBJECT23` which links them together.

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, $P\left(P_j|P_i\right)$, is given by the following formula:

$$P\left(P_j|P_i\right)=\frac{W_{P_i,P_j}}{\sum\limits_{k\neq i}W_{P_i,P_k}} \qquad \texttt{(F-1),}$$

where $W_{P_i,P_j}\geq 0.0$ is the strength between proposition nodes P_i and P_j. We describe how the strengths are computed in Section 3.3.2.5.

3.3.2 Stage of Execution

In simulation of the stage of execution, the program selects three candidate objects and then selects and executes an action-object pair. Each of these selection processes involves two phases: construction of a network of propositions and then use of an iterative, spreading activation process to make the selection. This section describes the network construction process, the integration process, selection of candidate objects, selection and execution of an action-object pair, and the parameters that specify the strengths of the links in the network.

3.3.2.1 The Network Construction Process

The simulation builds two networks during the stage of execution: one for candidate object selection and the other for action-object selection. The networks are represented by a square matrix where the rows and columns are propositions. The first is defined by the task and device goals, the display representation, the knowledge retrieved from long-term memory, and the representations of all candidate objects. The second is defined by the task and device goals, the display representation, the knowledge retrieved from long-term memory, and the representations of all actions on the three, selected candidate objects. If two propositions share one or more arguments, they are linked. The strength of link is determined by the parameters of the model described in Section 3.3.2.5 and summarized in Figure 5. These same link strengths are used to calculate retrieval probabilities, Eq. F-1, for the elaboration process. If there is no overlap between arguments of two propositions, there is no link, the strength is 0.0.

3.3.2.2 The Network Integration Process

The model selects the three candidate objects and the action-object pair to be executed using the integration process. On each iteration of the integration process, a vector of activation values is premultiplied by a matrix representation of the network. In the first iteration, the vector is set as follows: the elements for task and device goals, and display, which are *sources of activation*, are set to 1.0, and the elements for propositions retrieved from long-term memory, and candidate objects or representations of action-object pairs, are set to 0.0. The vector is renormalized after each iteration; the elements for sources of activation are reset to their initial values, 1.0, and the others are normalized so that their sum becomes a constant value. The iterative integration process stops when changes in the values of activation vector become below a threshold value.

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

3.3.2.3 Selection of Candidate Objects

The model first builds a network that links and assigns weights between propositions representing the task goal, the device goal, the elaborated display representation, and the propositions representing candidate objects. Then, the network is integrated. The model selects three candidate object representations with the largest activation values.

3.3.2.4 Selection and Execution of an Action-Object Pair

The model first constructs representations of action-object pairs for each of the three candidate objects. The representations of all action-object pairs are then linked in a network made up of propositions representing the task goal, the device goal, the elaborated display representation through the propositions that make up the first component of the action representation.

The representation of each action-object combination includes a set of conditions, the second component of the action representation, and consequences of performing the action on the object, the third component. The simulation puts additional links among the representations of action-object pairs that reflect causal relation between conditions and consequences. If a condition of an action-object combination, 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. Supports is represented in the network by a link with positive weight; inhibits is represented by a link with negative weight.

For example, let L_i be the pointing 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. These causal links are asymmetric.

The simulation also uses inhibitory links to prevent repeated execution of an action-object pair. The program examines for each action whether all consequences are found in the nodes representing the elaborated display representation. If they are found, the action is inhibited by all nodes that match the representation of consequences. These links are asymmetric.

	task-goal	device-goal	display	retrieved LTM	candidate objects	action-object pairs
task-goal	(1)	F_{goal}^2 $W_{overlap}$ W_{assoc} (1280)	F_{goal} $W_{overlap}$ W_{assoc} (80)	F_{goal} $W_{overlap}$ W_{assoc} (80)	F_{goal} $W_{overlap}$ (64)	F_{goal} $W_{overlap}$ (64)
device-goal		(1)	F_{goal} $W_{overlap}$ W_{assoc} (80)	F_{goal} $W_{overlap}$ W_{assoc} (80)	F_{goal} $W_{overlap}$ (64)	F_{goal} $W_{overlap}$ (64)
display			$W_{overlap}$ W_{assoc} (5)	$W_{overlap}$ W_{assoc} (5)	$W_{overlap}$ (4)	$W_{overlap}$ (4)
retrieved LTM				$W_{overlap}$ (4)	$W_{overlap}$ (4)	$W_{overlap}$ (4)
candidate objects					$W_{overlap}$ (4)	NA
action-object pairs						$W_{actions-excitation}$ $W_{actions-inhibition}$ (4), (–4)

Figure 5. A matrix representation of the network.

The network is then integrated and the most highly activated and eligible action-object pair is selected as the next action. The condition of an action-object pair is considered to be satisfied when all propositions in it’s condition are found in the elaborated display representation.

3.3.2.5 Model Parameters

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

$W_{overlap}$:	argument overlap weight,
W_{assoc}:	free association weight,
$W_{actions-excitation}$:	support weight,
$W_{actions-inhibition}$:	inhibit weight, and
F_{goal};	goal magnification factor.

The number of shared arguments, *argument overlap* , determines the strength of a link between a pair of propositions. It is assumed that, when two nodes share one argument, they are connected by a link of strength $W_{overlap}$. When they share N arguments, the strength is multiplied by N. This link is symmetric.

For example, a display representation, P1,

```
  Object23 IS_ON_SCREEN,
```

and a representation in long-term memory, P7,

```
  Object23 IS_ASSOCIATED_WITH Line-Graph-
    Menu-Item,
```

are connected by the shared argument, OBJECT23.

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_{asssoc}. For example, when P1 retrieves P7, the strength of the link between them becomes $W_{overlap} + W_{assoc}$. This link is symmetric.

The links between condition and consequences of action-object pairs take a value depending on whether or not the consequences of one pair support or inhibit another pair. The strengths of support and inhibit links are parameterized by $W_{actions-excitation}$ and $W_{actions-inhibition}$, respectively. These links are asymmetric.

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 goal magnification factor, F_{goal}, which is greater than or equal to 1. These special links have strong effects on the elaboration process, object selection, and action-object selection described more detail in 4.3.2. These links are symmetric.

Figure 5 shows a matrix representation of the network which is segmented into its constituents. Each cell describes relevant parameters for connecting two portions of the network. In order to illustrate rough shape of the matrix, a set of typical link strengths used in the simulation is also shown in the figure. The values in parentheses are link strengths for the case where there is one shared argument between two propositions, and where the link strength has been incremented by W_{asssoc} if one proposition is a retrieval cue for the other.

4. A SIMULATION EXPERIMENT

We have explored the behavior of the model in a series of simulation experiments. The initial studies, described in Kitajima and Polson (1994), were designed to understand how the model reacts to manipulations of its parameters. These studies are only summarized here. This section focuses on the results of a very large simulation experiment that was conducted in order to understand how and why the model makes errors. We adapted a program, NETWORK (Mannes and Roushey, 1990) to carry out our simulations.

4.1 The Initial Series of Simulation Experiments

The most startling result of our initial series of simulation experiments was that the model can fail to select correct actions even when it is provided with correct goals and all other information sufficient to generate the correct action sequence. We will show that the model does not guess when it makes an error, but it selects the most reasonable action based on incomplete information. Finally, we will demonstrate the model’s ability to recover from errors.

The initial series of simulation experiments were reported in detail in Kitajima and Polson (1994). The initial goal of these studies was to find a collection of parameter values for the model that would enable us to simulate skilled performance on a realistically complex task like the Cricket Graph Task. We assumed that skilled performance involved rapid generation of the correct action sequence. We found a set of parameters that caused the integration process to converge rapidly when generating the correct action sequence.

A major result of these experiments was that the links between task and device goals and the rest of the network had to be much stronger, 16 times, than any other links in the network in order for the model to generate the correct action sequence. We added a goal magnification factor parameter to the original Mannes and Kintsch (1991) model, F_{goal}. The consequence of this modification is that links between the goals dominate the processes involved in retrieval of information from long-term memory and candidate object and action selection. The links between the goals, the display objects, and the actions are the most important in the model because of the large value of F_{goal}.

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

Step No.	Task (**TG**) and Device goals(**DG**)	Correct Action
	<u>*Subtask 1*</u> **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*
	<u>*Subtask 2*</u> **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*

The simulation experiment with the Cricket Graph Task reported here explored the consequences of manipulating the elaboration parameter, N_{sample}. Based on the results of the initial simulation experiments, we used the following set of parameter values: $W_{overlap} = 4.0$, $F_{goal} = 16.0$, $W_{actions-excitation} = 4.0$, $W_{actions-inhibition} = -4.0$, and $W_{assoc} = 1.0$. The argument overlap parameter, $W_{overlap}$, is much larger than the value, 1.0, typically used by Kintsch (1988) in his simulation experiments. However, a value of 4.0 was necessary to produce rapid convergence of the integration process.

4.2 Method

Performing the Cricket Graph Task described in Section 1.4 was simulated in this experiment. The task was to reproduce the graph shown in Figure 1 using Cricket Graph on a Macintosh computer. Recall that 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.

The task and device goals and correct actions 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. There are seven major display changes in the correct sequence; before steps 1, 3, 4, 5, 7, 9, and 11. The details of the simulation are described in Section 3.3.

We did 50 simulation runs for each of the following values of N_{sample}; 4, 8, 12, 16, and 20. The remaining parameter values are listed in Section 4.1.

4.3 Results

4.3.1 Primary Cause of Errors

Our first series of analyses focused on the primary cause of an error. There are three possible causes. First, the model can fail to include the correct object in the set of candidate objects during the object selection process in the stage of execution. The second is that the correct action-object pair fails to become the highest activated pair among the executable action-object pairs during the action selection process. The third cause of an error is that the elaboration process fails to incorporate all of the conditions for the correct action-object pair in the elaborated display representation.

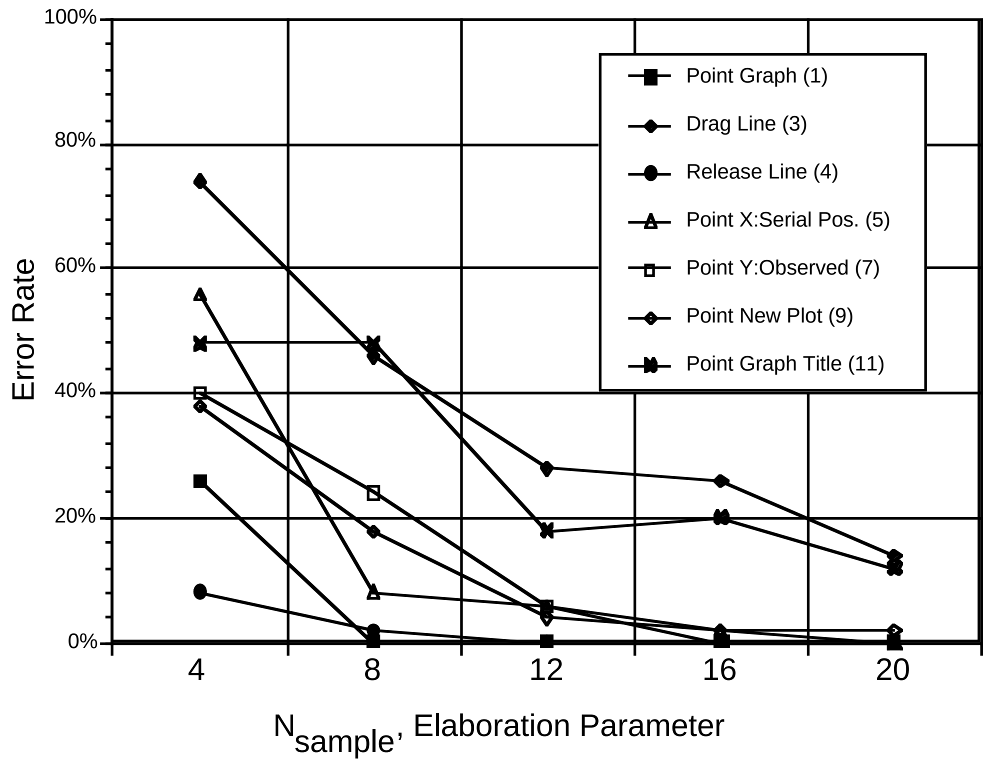


Figure 6. Probability of selecting correct action after major display change as a function of N_{sample}. All but release are actions to point at correct object.

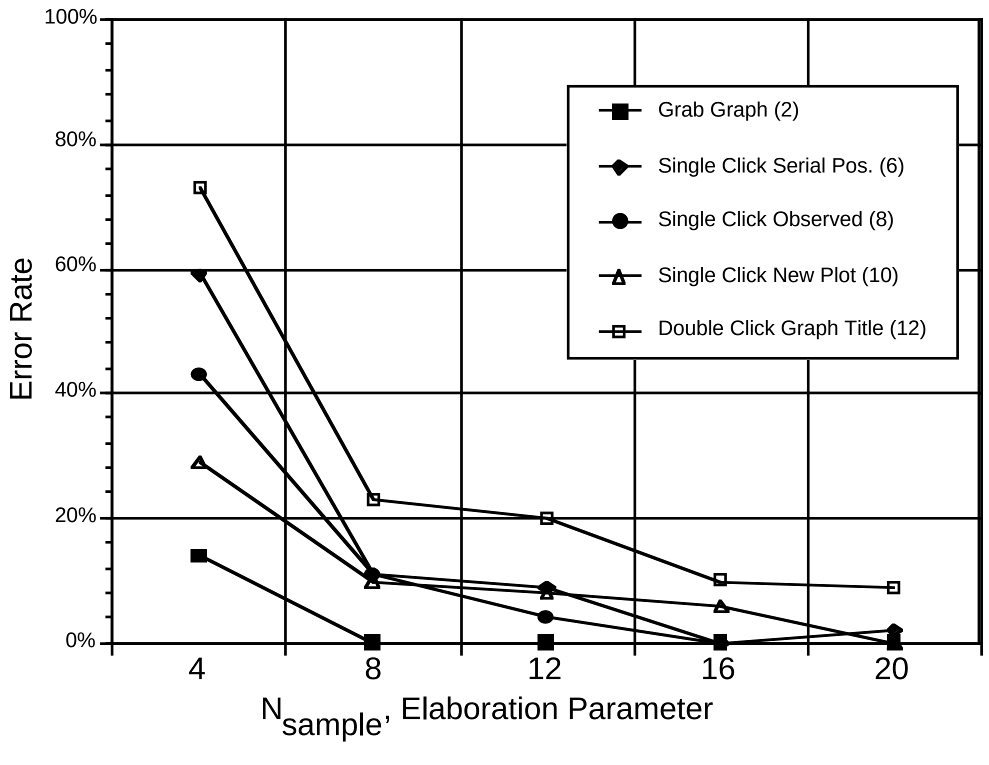


Figure 7. Probability of selecting correct action conditional on pointing at correct object.

The major results of these first series of analyses was that the first and second causes only generated one out of the 396 errors observed in 250 simulation runs of twelve steps of the task. Missing conditions was the sole sources of errors with one exception. The model never failed to select the correct candidate object over all steps and values of N_{sample}. There was only one activation failure in selecting the action, pointing at **Graph** on step 1 when $N_{sample} = 4$. These processes were not major causes of errors because they are primarily controlled by the very strong links between the correct task and device goals and the node representing the correct object and the correct action-object pair, respectively. The sole source of errors was failure of the elaboration process to include all of the conditions for the correct action in the network. In order words, the model predicts that most errors are description errors.

4.3.2 Error Rates by Step

Recall that the simulation executes all four of its major processes (input of goals and display representations, elaboration, candidate object selection, and action selection) and is given the correct goals and a representation of the display that results from the correct action after each major display change, steps 1, 3, 4, 5, 7, 9, and 11. The actions all involved moving the mouse cursor except for step 4 which was to release on the menu item **Line-Graph**. We have partitioned the following analyses into two sets, the first defined by these steps. The second set are the actions on steps 2, 6, 8, 10 and 12. This second set of analyses describes what happened after both correct and incorrect actions on the preceding step.

Figure 6 plots the error rates on steps 1, 3, 4, 5, 7, 9, and 11 as a function of N_{sample}. Figure 6 shows that error rates decreases as a function of N_{sample} but that the initial values at $N_{sample} = 4$, and the rates of decrease differ as a function of step. For example, pointing at **Graph** in the menu bar on step 1 had an error rate of 26% for $N_{sample} = 4$, and fell to 0% at $N_{sample} = 8$. Pointing at **Line-Graph** in the **Graph** menu, step 3, had an error rate of 74% for $N_{sample} = 4$ and decreased to 14% when $N_{sample} = 20$.

Figure 7 shows the error rates for the steps 2, 6, 8, 10, and 12, conditional on the correct preceding action selection. Again, the error rates decrease as a function of N_{sample} but that the initial values, and the rates of decrease as a function of step differ across steps. Step 2 was easiest; steps 6, 8, and 10 were of moderate difficulty; and step 12 was the hardest.

The results of the simulation shows that pointing at **Graph** in the menu bar and grabbing it (steps 1 and 2) and releasing on **Line-Graph** in the **Graph** pull down menu (step 4) were the easiest steps. Interactions with the variables selection dialog box (steps 5 through 10) were of intermediate difficulty. Pointing at the **Graph-Title** and double clicking it (steps 11 and 12) were hard. Pointing at **Line-Graph** in **Graph** pull down menu was the hardest step. This section describes the reason why error probability differs from steps.

The simulation results show that memory retrieval failures during the elaboration process are the cause of errors made by the model. The retrieval failures cause propositions to be omitted from the network required to satisfy the conditions of the correct action. Kitajima (1995) has shown that the differences in error rates are due to the number of conditions in the correct action that must be retrieved from long-term memory and the probability that a specific condition will be retrieved.

Pointing at **Line-Graph** in **Graph** pull down menu was the hardest step because three propositions had to be retrieved from long-term memory in order to satisfy the conditions of this action.

The correct actions for the remaining 11 steps has just one condition that had to be retrieved from long-term memory. The difficulty of these steps shown in Figures 6 and 7 are determined by differences in the retrieval probabilities, calculated using formula F-1 (Raaijmaker & Shiffrin, 1981).

In order to understand how retrieval probabilities are determined by F-1, we need to consider several special cases. This formula is the ratio of the strength of the link between the retrieval cue and the proposition to be retrieved from LTM divided by the sum of the strengths of the links between the retrieval cue and all propositions that link to the retrieval cue. The probability of retrieval is determined by the number and strength of these competing associations.

Let P_{LTM} be the proposition representing the one condition of the correct action. See Figure 8. Suppose that there is one cue, P_{disp}, in the display that links to P_{LTM} by a single shared argument. In the first case, P_{disp} has linkages with $P_{disp'}$, $P_{disp''}$, P_{LTM}, and $P_{LTM'}$, but neither with P_{task}, the task goal, nor with P_{device}, the device goal. In this case, the probability for P_{disp} to retrieve P_{LTM} in a single memory retrieval is 0.25. The second case is where P_{disp} is linked to P_{device} in addition to $P_{disp'}$, $P_{disp''}$, P_{LTM}, and $P_{LTM'}$. In this case, the probability for P_{disp} to retrieve P_{LTM} is reduced significantly to 0.05 $= 1/(16+4)$. The third case is where P_{disp} is linked to both P_{task} and P_{device} in addition to $P_{disp'}$, $P_{disp''}$, P_{LTM}, and $P_{LTM'}$. The probability is only 0.028 $= 1/(16+16+4)$.

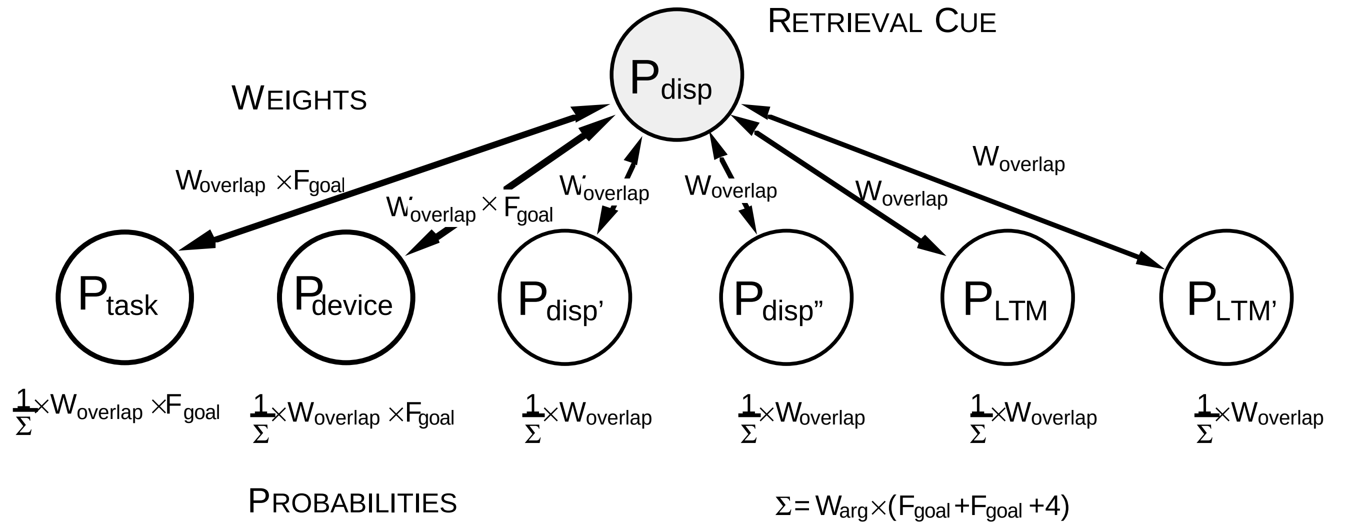


Figure 8. Explanation of mechanism of potential retrieval failure due to interactions between the goal magnification factor parameter, F_{goal}, and the memory sampling process.

Although the absolute values of probabilities that the condition proposition is retrieved from long-term memory in the simulation are different from the above values, the significant effects of powerful links between the cue proposition and the goals on the retrieval probabilities of the condition proposition are maintained. On steps 1, 2, and 4, the retrieval cues for the condition proposition had no linkage with the goals, which correspond to the first easiest case. All actions involving the variable selection dialog box correspond to the second case. Steps 11 and 12 are the example of the third case.

4.3.3 What The Model Does When It Makes An Error

In the last section, we showed that error rates range from 0% to 70% as a function of steps and N_{sample}. Errors were caused by failures of the elaboration process to include in the network a necessary condition for the correct action. However, when the model makes an error, it does not respond randomly. Recall that the model can only execute an action on one of the three candidate objects. The model described in this paper makes errors by performing a possible action on one of the two incorrect candidate objects. The underlying architecture also permits incorrect actions on the correct object.

In the first step of the task, the correct action is to point at **GRAPH** on the menu bar. For 50 simulation runs where $N_{sample} = 4$, the model correctly pointed at **GRAPH** 37/50 times. On 20% of the simulation runs, it pointed at **EDIT**. It pointed once at the highlighted cell in the spreadsheet shown in Figure 2, and it pointed twice at **FILE** in the menu bar. On step 2, the model grabbed the pointed at object 45/50 times. The model pointed at some other object on the five remaining simulation runs. When the model was interacting with the variable selection dialog box on steps 5 to 10, all errors involved either pointing at or single clicking on a wrong object in the dialog box.

In summary, errors made by the model are strongly constrained. When the correct action is "move the mouse pointer to a specified object", the most frequent error is to point at a wrong object. Once the model has pointed at an object, it very frequently executes the action on this next step that would be correct if it pointed at the correct object on the previous step.

This behavior of the model is caused by the very different ways that the candidate object and action selection processes react to variations in the elaboration parameter, N_{sample}. The candidate object selection process is unaffected by changes in N_{sample}. With one exception, the model included the correct object among the candidate objects in all simulations of each step. Furthermore, the model selects objects related to the task and device goals for the other two incorrect candidate objects.

The action selection process is very sensitive to reductions in N_{sample}. Missing necessary conditions for the correct action block its execution. However, a wrong action will be related to the correct action because the action selection process is dominated by the links between possible actions and the goals. The wrong objects and the actions on those objects are closely related to the goals which means they are closely related to the correct actions on the correct object.

4.4 Recovery From Errors

In this section, we describe a small simulation that shows that the candidate object selection process in the stage of execution enables the model to recover from errors. In this experiment, we simulated the task of editing the **Graph-Title**, steps 11 and 12. The correct actions are pointing at the **Graph-Title** followed by double clicking it. Here again, the behavior of the model is determined by the difference in how the candidate object and action selection processes react to variations in N_{sample}.

We focused on the error pointing at the **Edit** menu followed by pulling it down. Upon the detection of this event, the model was given the display defined by the items on the **Edit** menu appearing on the screen. The model then executed all four processes in the action cycle. In the candidate object selection process, the model selected three candidate objects which did not include *any of* the **Edit** menu items. There were no links between the task and the device goals and any of the items on the **Edit** menu. However, the model included **Graph-Title** on the list of the candidate objects. In addition, the model successfully retrieved the propositions representing the conditions for the correct actions during the elaboration process. The model was then able to select the action of pointing at the **Graph-Title** and then double clicking it.

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, it would have been selected as a candidate object. The model could have continued on this erroneous course of action .

5. DISCUSSION

5.1 The Mannes and Kintsch (1991) Model

Kitajima and Polson (1992, 1994) 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. Mannes and Kintsch (1991) added a new construct to Kintsch's (1988) construction-integration model, called the plan element.

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.

5.2 Extensions

Our model extends previous models of human-computer interaction using the construction-integration framework in four ways.

5.2.1 Errors

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 (Kitajima and Polson, 1992, 1994). Our major result, 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%). Furthermore, the model can explain some forms of error recovery. 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 $N_{sample} \approx 12$.

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 Kintsch's (1988) 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 of the comprehension process. It turns out, especially for text comprehension, that the resulting representation is redundant and therefore quite robust.

Our model just simulates action slips (Norman, 1981; Reason, 1990). Recall that it is given the correct goals for the task. It can account for the observed error rates in skilled performance as well as the fact that action slips generate errors that are closely related to the correct action (e.g., Reason, 1990). The ability of the model to simulate slips as well as some simple forms of error recovery are do to the more complex action selection mechanism incorporated into our model.

Action selection during the stage of execution involves two process: 1) selection of three candidate objects, and 2) the selection of the correct action object pair from a set of all possible actions on the three candidate objects. These processes react very differently to variations in N_{sample}. The first process is unaffected by variations in N_{sample} and always includes the correct object and two other related objects in the set of candidate objects. This results in errors that are related to the correct action. The second is quite brittle. The model makes errors if N_{sample}, the elaboration parameter, is not set to a large value, causing conditions necessary for execution to be omitted from the network. 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. Our model claims that experts' errors are slips caused by failure to generate complete representations of objects on the screen.

5.2.2 Selection of Candidate Objects

Second, the model has a more detailed representation of information contained in realistically complicated displays. The environment we simulated, 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.

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.

5.2.3 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 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.

5.2.4 Grain Size of Actions

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, 1994) 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.

6. 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. (1986) 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 several extensions to the Mannes and Kintsch (1991) model of action planning. The model simulates an environment in which there is a complex display with many irrelevant objects. The 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. We modified the Mannes and Kintsch action representation. We assumed a richer structure on actions incorporating intentions into the action representation. 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.

The most important set of theoretical results was to show that the model provides a principled explanation of errors made by skilled 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. (1992) 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 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 for the novices will tend to be highly error prone for the experts.

7. ACKNOWLEDGMENTS

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. Peter Polson was support by a National Research Council Senior Research Associateship during the writing of the final version of this paper. 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. The opinions expressed in this paper are those of the authors and not necessarily those of any of the supporting organizations. 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.

8. 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.

KINTSCH, W. & MROSS, E.F. (1985) Context effects in word identification. *Journal of Memory and Language*, **24**, 336-349.

KITAJIMA, M. (1995). Action selection error probabilities in the construction-integration theory-based model of skilled use of display-based HCI. *ICS Technical Report*. 95-XX. Boulder, CO: Institute of Cognitive Science, University of Colorado.

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.

KITAJIMA, M. & P OLSON, P. G. (1994). A comprehension-based model of correct performance and errors in skilled, display-based human-computer interaction. *ICS Technical Report*. 94-02. Boulder, CO: Institute of Cognitive Science, University of Colorado.

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.

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 $i \neq j$;

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

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

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

$$\times \begin{Bmatrix} F_{goal} \text{ ; if } P_i \text{ or } P_j \in \{\text{device goals}\} \\ 1.0;\ \text{otherwise} \end{Bmatrix}$$

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, $L_k \in$ {actions},

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

$$W_{overlap} \times (\text{number of shared arguments})$$

$$\times \begin{Bmatrix} F_{goal} \text{ ; if } P_i \in \{\text{task goals}\} \\ 1.0;\ \text{otherwise} \end{Bmatrix}$$

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

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

$$W_{P_i,L_k} = W_{actions-inhibition}$$

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

For $k \neq l$;

$$W_{L_k,L_l} = \begin{cases} W_{actions-excitation};\ \text{if } L_k \text{ supports } L_l \\ W_{actions-inhibition};\ \text{if } L_k \text{ inhibits } L_l \\ 0.0;\ \text{otherwise} \end{cases}$$

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

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

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 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 ν-th flash can be expressed by a vector, $\vec{\mathbf{A}}^{(\nu)}$, and the strengths in the network, by a matrix, $\mathbf{W}$, as follows, respectively;

$$\vec{\mathbf{A}}^{(\nu)} \equiv \left(\vec{\mathbf{A}}_S^{(\nu)}, \vec{\mathbf{A}}_R^{(\nu)}\right);$$

$$\vec{\mathbf{A}}_S^{(\nu)} \equiv \left(A_{S_1}^{(\nu)}, \cdots, A_{S_N}^{(\nu)}\right)$$

$$\vec{\mathbf{A}}_R^{(\nu)} \equiv \left(A_{R_1}^{(\nu)}, \cdots, A_{R_{N'}}^{(\nu)}\right)$$

$$\mathbf{W} \equiv \begin{bmatrix} W_{S_1,S_1} & \cdots & W_{S_1,S_N} & W_{S_1,R_1} & \cdots & W_{S_1,R_{N'}} \\ \cdots & & \cdots & \cdots & & \cdots \\ W_{S_N,S_1} & \cdots & W_{S_N,S_N} & W_{S_N,R_1} & \cdots & W_{S_N,R_{N'}} \\ W_{R_1,S_1} & \cdots & W_{R_1,S_N} & W_{R_1,R_1} & \cdots & W_{R_1,R_{N'}} \\ \cdots & & \cdots & \cdots & & \cdots \\ W_{R_{N'},S_1} & \cdots & W_{R_{N'},S_N} & W_{R_{N'},R_1} & \cdots & W_{R_{N'},R_{N'}} \end{bmatrix}$$

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

The initial activation values for the source nodes, $A_{S_i}^{(0)}(\text{for } 1 \leq i \leq N)$, are set to 1.0, and for the receptor nodes, $A_{R_j}^{(0)}(\text{for } 1 \leq j \leq N')$, to 0.0. The activation vector after ν-th activation flash, $\vec{\mathbf{A}}^{(\nu)}$, is defined as follows;

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

$$A_{S_i}^{(\nu)} = 1.0$$

For receptor nodes,

$$A_{R_i}^{(\nu)} = N \times \frac{\max(0.0, \tilde{A}_{R_i}^{(\nu)})}{\sum_{k=1}^{N'} \max(0.0, \tilde{A}_{R_k}^{(\nu)})}$$

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

$$\tilde{A}_{R_i}^{(\nu)} = \sum_{j=1}^{N} A_{S_j}^{(\nu-1)} \times W_{S_j,R_i} + \sum_{j=1}^{N'} A_{R_j}^{(\nu-1)} \times W_{R_j,R_i}$$

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

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;

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

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.