A Comprehension-Based Model of Exploration

Muneo Kitajima
National Institute of
Bioscience and Human-Technology
1-1 Higashi Tsukuba Ibaraki 305, JAPAN
Tel: +81 (298) 54-6730
E-mail: kitajima@nibh.go.jp

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

ABSTRACT
This paper describes a comprehension-based model of how experienced Macintosh users learn a new application by doing a task presented as a series of exercises. A comprehension mechanism transforms written instructions into goals that control an action planning process proposed by Kitajima and Polson [11]. The transformation process is based on a theory of solving word problems developed by Kintsch [8,9]. The comprehension and action planning processes define constraints on the wording of effective instructions. The combined model is evaluated using data from Franzke [3]. We discuss implications of these results for Minimalist Instructions [1] and Cognitive Walkthroughs [17].

Keywords
cognitive theory; display-based systems; exploration

INTRODUCTION

The goal of this article is to describe the LICAI[1] model, which extends the Kitajima and Polson [11] action planning model of skilled, display-based, human–computer interaction to account for learning by exploration. Learning by exploration involves discovering how to do a novel task by generalizing from past experience or by searching successfully for a correct action sequence. The LICAI model describes how experienced Macintosh users accomplish a task using a new application by doing a series of exercises. These users are familiar with the standard Macintosh interface conventions. They must combine their existing knowledge of the interface conventions with the task described in each exercise to generate the actions required by the new application to perform each task.

The LICAI model incorporates processes from Kintsch's [8,9] theory of text comprehension. The LICAI model uses comprehension strategies to transform written instructions into goals. In this paper, we propose three comprehension strategies in the form of schemata. Schemata are specialized knowledge structures whose slots are filled with crucial elements extracted from the instructions. The output of the comprehension processes are goals that control the action generation processes.

The LICAI model uses the action planning processes developed by Kitajima and Polson [11]. Their action planning model can simulate the behavior of skilled users interacting with a graphing application hosted on the Macintosh. The comprehension-based model of goal formation processes and the model of action planning processes are combined to describe the behavior of experienced Macintosh users learning a new application. The LICAI model defines constraints on the wording of effective instructions.

The LICAI model is evaluated using data from Franzke [3], who had experienced Macintosh users do a graphing task using one of three different graphing applications with which they had no prior experience. The graphing task was presented to users as a series of exercises. The instructions for each exercise contained no information about the action sequences required to complete each subtask. Franzke's [3] experimental task is similar to actual learning and use of an application. Users formulate tasks for themselves or are given written specification for a new task.

Franzke [3] found large variations in difficulty of subtasks both within and across different applications. We argue that Franzke's [3] participants had to comprehend the goal of each exercise and then infer, or try to discover by exploration the action sequence that would accomplish the goal. The goal formation processes attempted to build specialized goals required by Kitajima and Polson's action planning model [11] to generate correct actions for a subtask. These goal structures provided links between

[1] LICAI is acronym for the **LI**nked model of **C**omprehension-based **A**ction planning and **I**nstruction taking. When LICAI is pronounced like "Lee CHI (χ)," the pronunciation represents the two-kanji character Japanese word, 理解, which means comprehension.

users' understanding of a subtask and the low-level details of the interface to the new application, (e.g., menu labels).

Building these linking goal structures is a difficult and highly specialized comprehension task, analogous to processes required to successfully solve word problems studied by Kintsch [8,9]. The goal formation processes can fail. Users may not have all of the necessary comprehension strategies and/or knowledge about the task or about the application interface. Our theoretical analysis of Franzke's [3] results refines and extends her analyses, which were based on the model of learning by exploration [17] underlying the Cognitive Walkthrough [18,19].

Previous Research

The LICAI model characterizes skilled performance as involving comprehension processes that map an initial task description onto a specialized problem-solving representation. Greeno and Simon [4] showed that such mapping processes are a recurrent theme in the problem-solving literature. A good example is Hayes and Simon's [5] model of solving simple laboratory problems (e.g., tower of Hanoi and water jugs) described to participants in written instructions. Their model took as input the problem text and mapped it onto the representation required by the General Problem Solver [2] to solve the problem. Howes and Young [6] developed a SOAR model that analyzed interactions among the content of instructions given during learning by exploration, the details of the interface, and users' background knowledge.

Outline of the Paper

The next section describes the theoretical foundations and the action planning model of skilled, display-based human–computer interaction developed by Kitajima and Polson [11] and summarizes the results of a series of simulation experiments reported by Kitajima [10]. We then describe our comprehension-based model of goal formation that maps an initial description of a task onto the specific goals required by Kitajima and Polson's [11] action generation process. We evaluate the LICAI model using Franzke's [3] results. Finally, we summarize the implications of our results for practice. These results have importation implications for the design of interfaces and training materials that support learning by exploration [18] [1].

ACTION PLANNING MODEL OF DISPLAY-BASED HUMAN–COMPUTER INTERACTION

Kitajima and Polson's [11] model of action planning is synthetic in that it attempts to integrate the views of numerous researchers on the nature of display-based, human–computer interaction (e.g., [7]), theoretical ideas about the nature of display-based problem-solving (e.g., [13]), action planning [15], and task and device representations [16].

There are two core ideas underlying the action planning model. First, Kitajima and Polson [11] proposed that display-based HCI is analogous to *text comprehension*. In reading text, readers use large amounts of knowledge to comprehend the meaning of texts. In display-based HCI, users must comprehend the display and then select appropriate actions with the help of knowledge about the interface, the task to be performed, and so on. Second, the model is mapped onto Hutchins et al.'s [7] analysis of direct manipulation based on their action theory framework. This framework describes action planning as a goal driven process that evaluates the consequences of the last action and then generates the next action to be executed (see Figure 1).

Display-Based HCI and Text Comprehension

The action planning model implements a version of the action theory framework by extending Mannes and Kintsch's [15] theory of action planning, which is based on Kintsch's [8] construction–integration model of text comprehension. Kintsch proposed a model of comprehension that combines elements of symbolic and connectionist models of cognitive processes. Text comprehension is a cyclic process where readers process a sentence, or the major constituent of a longer sentence, during a single cycle; reading a text involves a sequence of such cycles. On each construction–integration 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. Kintsch's model outputs a representation of this latest sentence or fragment consistent with the reader's goals and the context provided by the previous text.

The construction–integration cycle is a two-phase process. In the first phase, a network of propositions is created that contains 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, at the end of the construction process, the model has multiple possible meanings for the input text.

The *integration process*, the second phase, selects 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 the reader's interpretation.

Mannes and Kintsch [15] extended the construction–integration theory to action planning. Their task domain was human–computer interaction. Their action-planning model took as input a representation of users' or planners' goals, a propositional representation of the text containing the task description, and a very schematic representation of the task context. Their model generated the commands required to perform the task described in the text. Mannes and Kintsch argued that text comprehension and action planning can be conceived 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.

Kitajima and Polson's Model of Action Planning

Kitajima and Polson [11] developed an action planning model of display-based HCI based on Mannes and Kintsch's [15] construction–integration model of action planning, mapped onto the action cycle theory framework of Hutchins et al. [7]. The outline of the Kitajima and Polson model is shown in Figure 1.

Example Task

We will describe the action planning model by tracing its behavior during part of the task involving drawing a line graph for data contained in a table with columns labeled with variable names. The example plotting task is represented as follows:

Plotting-task: Plot numbers in the column labeled 'Observed' as a function of numbers in the column labeled 'Serial Position' (1)

Users first select a graph type from a pull-down menu, causing the dialog box in Figure 2 to appear. The column labels are displayed in two scrolling lists. Note that the label Serial Position appears in both X and Y axis scrolling lists. The dialog box partially occludes the table, but the column of numbers labeled Serial Position is visible in the background. To plot Observed as a function of Serial Position, users must click on and highlight Serial Position in the X-axis scrolling list and Observed in the Y-axis scrolling list. Finally, users point at the button New Plot and single-click it.

Goals

Kitajima and Polson's [11] action planning model assumes that skilled users have a schematic representation of the task in the form of a hierarchical structure involving two kinds of goals: task goals and device goals [16]. They assumed that each task goal is associated with one or more device goals. The device goals specify device states that must be achieved to satisfy an associated task goal. When the action planning model is provided with a new display, the model retrieves task and device goals. The task and device goals were:

Task goal: Perform "Put Serial Position on the X-axis" (2)

Device goal: Realize "Serial-Position-in-X-axis-scrolling-list is-highlighted" (3)

The Evaluation Stage

The action planning model is given a representation of a new display in the form of a large collection of screen objects; each screen object is described by several propositions. These descriptions include only limited information about the identity of each object and its appearance, including visual attributes (e.g., color, highlighting).

The model simulates Hutchins et al.'s [7] evaluation stage (shown in Figure 1) by elaborating the display representation with knowledge retrieved from long-term memory. The retrieval cues are the task and device goals and the propositions representing the current display. The

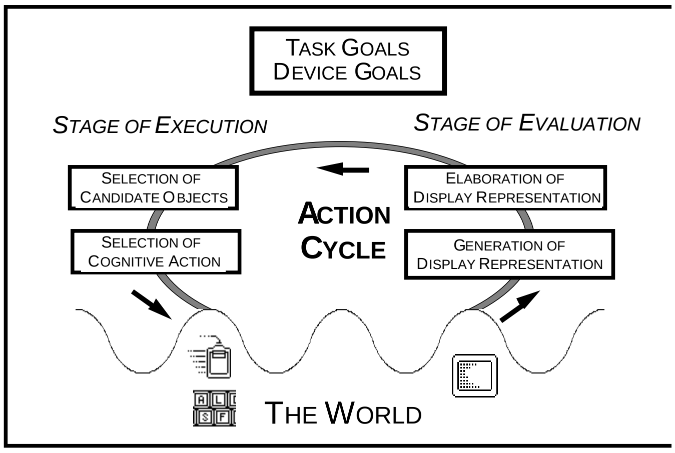
TASK GOALS
DEVICE GOALS
STAGE OF EXECUTION
STAGE OF EVALUATION
SELECTION OF CANDIDATE OBJECTS
ELABORATION OF DISPLAY REPRESENTATION
ACTION CYCLE
SELECTION OF COGNITIVE ACTION
GENERATION OF DISPLAY REPRESENTATION
THE WORLD

Figure 1. The action planning model of correct performance and errors in skilled, display-based HCI [11].

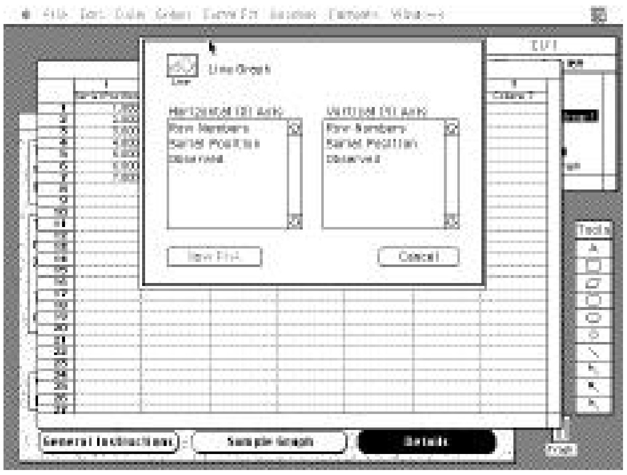

Figure 2. Example task.

probability that a cue retrieves a particular proposition representing a piece of knowledge in long-term memory is proportional to the strength of the link between them. The propositions in long-term memory represent knowledge about the screen objects. For example, if Object23 is the scrolling list item labeled Serial Position, then the following knowledge items are stored in long-term memory about Object23: Object23 *has-label* Serial Position; Object23 *is-a-member-of* Line-Graph-Dialog-Box; Object23 *can-be-pointed-at;* Object23 *can-be-selected.* The elaboration process is stochastic and is taken from Kintsch [8]. Kitajima and Polson [11] discussed in detail the predictions and implications that follow from this stochastic elaboration process.

The Execution Stage

The execution stage of Hutchins et al.'s framework [7] is modeled by two construction–integration cycles. The first construction–integration cycle selects three screen objects as possible candidates for action. An important feature of the Kitajima and Polson's [11] action planning model is that the display representation is a detailed description of an actual large format display. Thus, the model's display representation can incorporate up to 100 screen objects. All screen objects are candidates for possible actions. During the initial construction phase, representations of all screen objects are combined with the goals and the elaborated display representation to construct the network. When the integration process converges, the model selects the three most highly activated screen objects as candidates for the next action.

This process is dominated by two factors. First, strong links from the goals to propositions in the network that share arguments with the goals, and second, the number of propositions necessary to link goals to candidate objects. As a result, the action planning model selects candidate objects closely related to the task and device goals. Device goals can directly specify a screen object, and thus can be directly linked to the screen object represented in the network. Task goals can be linked to screen objects through labels. Thus, the task goal shown in (2) is linked to the object representing the variable Serial Position in the X-axis scrolling list by the overlap of the labels that are part of the display representation.

The second construction–integration cycle selects an action to be performed on one of the three candidate objects. During the construction phase of this second cycle, the model generates a network with representations of all possible actions on each candidate object. Examples would include single-clicking and moving the screen object labeled Serial Position in the X-axis scrolling list. At the end of the second integration phase, the action planning model selects the most highly activated object–action pair as the next action to be executed. The process is dominated by the same two factors described above. However, the relevant interaction knowledge must be retrieved during the evaluation stage. For example, the action planning model must retrieve the fact that objects in the scrolling list can be selected.

Goal Specificity

Kitajima and Polson [11] assumed that they were modeling skilled users of the graphing package, and they gave the action planning model the very explicit goals required, examples of which are shown in (2) and (3). In a series of simulation experiments reported by Kitajima [10], we explored whether the model could successfully perform tasks with vague, incomplete, or even missing goals. Our initial conjecture was that users of a new application would have some understanding of the tasks they were going to perform, and they would be able to formulate more-or-less complete task goals. However, new users would not be able to formulate the precise device goals required, because they had never interacted with the application before.

The Kitajima [10] simulations focused on the two actions related to the task and device goals given in (2) and (3). The correct action sequence involved moving the mouse cursor to point at the label Serial Position in the X-axis scrolling list, followed by single-clicking on that object. The simulation experiments started with a display representation defined by Figure 2, and then the model attempted to perform the sequence of correct actions.

Kitajima [10] found that the action planning model can reliably generate the correct action sequence with no device goal. However, the task goal had to be stated exactly as given in (2). A perfectly reasonable task goal like "Plot Observed as a function of Serial Position" does not work. Kitajima [10] concluded from his simulations that the task goal had to be directly linked to the labels for the X-axis scrolling list and for the correct object in that scrolling list.

Furthermore, Kitajima [10] found an effect of number of competing screen objects. If the model was required to make the correct actions with a screen representation that included both the dialog box as well as the data table in the background with the distracting label Serial Position, these additional distracting objects prevented the action planning model from successfully generating the correct action sequence. However, limiting the focus of attention to the nine screen objects defined by the dialog box shown in Figure 2 enabled the specific task goal shown in (2) to generate the correct action sequence.

The action planning model always performed the correct actions given the device goal (3). The direct link between the device goal and the correct screen object caused the model to include the correct screen object in the list of the three candidate screen objects during the first phase of the execution stage. During the second phase, when selecting the correct action, information retrieved from long-term memory enabled the model to decide that the only possible action was to single-click on this object. The action planning model performed the task perfectly because the correct action was the only possible action.

A COMPREHENSION-BASED MODEL OF GOAL FORMATION

Kitajima's [10] results show that we can extend Kitajima and Polson's [11] original action planning model to account for people with a lot of background experience learning a novel application by describing how they formulate very specific task and device goals. Our goal-formation model assumes that this process is analogous to solving word-problems. The text comprehension processes take a semantic representation of the next task as input and combine this representation with highly specialized background knowledge to generate the required task goals. Device goals are acquired by interacting with the interface.

Problem Schemata for HCI

Task and interface specific problem schemata guide the transformation of the semantic representation of the original problem description and experiences of interacting with the program into a useful problem model (i.e., correct task and device goals). Problem schema is a notion proposed by Kintsch and Greeno [9] and further amplified by Kintsch [8]. A schema is a knowledge structure that takes a semantic representation of instructions as input and generates one or more specialized propositions defined by a predicate and slots with strong constraints on the admissible arguments.

The original problem statement "Plot Observed as a function of Serial Position" is transformed by two task specific problem schemata associated with the task Plot:

Put *variable-label$_1$* on X-axis (4)
Put *variable-label$_2$* on Y-axis (5)

In addition, specialized comprehension knowledge incorporated into the schema is required to fill the slots (i.e., "as a function of" means that the variable label before the phrase is put in the Y-axis slot, and the variable after the phrase is put in the X-axis slot). In the following section, we propose problem schemata that map instructions into task and device goals.

Schemata for Task Goal Formation

We assume two general schemata for task goal formation. The schemata construct propositions of the form (perform *action object*) which are used as task goals to generate associated actions.

TASK Schema

The TASK schema takes the original task instructions and transforms them into one or more propositions representing a task goal. This set of propositions is of the form, "perform *task-action* on *task-object* with *additional task-specification*," where both task-action and task-objects are constrained to be concepts for the task-domain. For example, transforming (4) generates the following instance of a TASK schema:

TASK schema	
task-action:	put
task-object:	Serial Position
task-specification:	on X-axis

The resulting task goal description is represented by two propositions: (perform put Serial_Position) and (location-of Serial_Position on_X-axis). Kitajima and Polson's [11] action planning processes then generates the sequence of actions that achieve this task goal.

The transformations performed by the TASK schema can be complex. The original task instructions can contain information irrelevant to the task goal, and thus the TASK schema must summarize the instructions to generate a task goal. The transformation shown in the above example is a simple paraphrase into a form that links directly to the labels of screen objects defined by the interface. The TASK schema can also generate necessary elaborations of terse instructions.

DO-IT Schema

In Franzke's [3] experiment, if participants were not making any progress on a subtask, they were given hints like "Click on Serial Position in the X-axis scrolling list." Observe that following such an instruction involves nontrivial inferences. 'Click on' must be mapped onto the action: Single-click with the mouse button after moving the mouse cursor to the required screen object. Serial Position must be mapped onto the screen object with the label Serial Position that is in the X-axis scrolling list.

The DO-IT schema maps instructions that describe a single legal action for the interface on a screen object with various attributes into a description of the form "perform *device-action* on *device-object* with *additional device-specification*." For example, the instruction, "Click on Serial Position in the X-axis scrolling list," is transformed into the following instance of the DO-IT schema:

DO-IT schema	
device-action:	single-click
device-object:	$; *variable undefined*
ID:	scrolling-list-item
attribute	
label:	Serial Position
location:	X-axis scrolling list

The resulting task goal is the following set of propositions: (perform single-click $), (isa $ scrolling-list-item), (has-label $ Serial_Position), (location-of $ X-axis_scrolling_list). Observe that this task goal specifies a single action. It's arguments link to the screen representation and action representation. The action planning processes can generate the specified step.

Schema for Device Goal Formation

It is unlikely that new users of an application would generate descriptions of a novel task that could be transformed by a schema into a device goal. Generating such descriptions requires detailed knowledge of the interface that can be only obtained by interacting with the particular application. For example, to be able to infer the device goal shown in (3) from the task description (1), users must know that a screen object labeled by the variable to be placed on the X-axis will be displayed as a scrolling list item and that this scrolling list item should be highlighted.

The DEVICE Schema transforms experiences interacting with the interface into device goals. The schema generates one or more propositions of the form "realize *device-object is-in-device-state* with *additional device-specification*." The result is a device goal like (3). For example, the experience of successfully highlighting Serial Position in the X-axis scrolling list generates the following instance of the DEVICE schema:

DEVICE schema	
device-object:	Object23
ID:	list-item
attribute	
label:	Serial Position
location:	X-axis scrolling list
display-state:	highlighted
associated-task-goal:	perform "put Serial Position on X-axis"

Experienced users employ the current task goal and / or display as a cue to retrieve device goals from long-term memory.

EVALUATION OF THE LICAI MODEL

In this section, we evaluate the LICAI model using data from Franzke [3]. Experienced Macintosh users were given the task of creating a new graph with a novel graphing application, Cricket Graph I[2] or III[3], or one of two forms of the EXCEL 3.0[4] interface. The graphing task was divided into two subtasks. The first was to create a default line graph by opening a document containing the data to be plotted, selecting the correct graph style (e.g., line graph) from a menu, and assigning the designated variables to the X- and Y-axis. The second subtask was to edit the default line graph. The edits were done in a specific order. The descriptions of the edits were very terse. Participants learned to do subtasks by exploration. If they had not made any progress toward the next correct action for more than 2 minutes on a particular step, they were given brief hints like "select line graph from the graph menu," or "double-click on legend text."

The goal formation processes of the LICAI model have been simulated using Franzke's [3] experimental paradigm [12]. The LICAI model only makes course grain predictions about the behavior of Franzke's participants. The instructions and the schemata assumed by the model may or may not enable participants to generate the correct task goal. If they generate the correct task goal, the action planning processes will generate the correct action sequence. However, the LICAI model does not describe the search behavior that occurs if the task goal construction process fails. We account for the initial success or failure of the goal formulation process. If instructions for a given exercise contain the necessary information, the comprehension processes will generate the goals that enable the action planning processes to generate the correct action sequence for the exercise. Thus, the LICAI model partitions the exercises given to Franzke's participants to tasks that can be done with little or no trial-and-error search and those that the model cannot perform because it can't generate the necessary task goal from the instructions. However, the model is able to generate a qualitative account of Franzke's results.

Label Following
Franzke [3] found strong support for the label following strategy [17,18] (see Figure 5 in Franzke [3]). Participants used overlap between task descriptions contained in instructions with labels on menus, buttons and other interface objects when learning by exploration. The degree of success of the label following strategy depended on the quality of the label and on the number of competing screen objects. Franzke [3] (Figure 7) obtained an interaction between number of objects (2 – 10) on the screen and quality of the label match (good, poor). There was no effects of number of objects for good labels and a large effect for poor labels. A unique, good label caused the user to attend to the correct screen object independent of the number of competing screen objects.

Label following is consistent with the LICAI model. If instructions contained a description of either an action or an object that matched a screen object label, those labels were preserved when the propositional representation of the instructions was mapped into a task goal. The links between the task goal and the correct screen object can mediate performance of the correct action if participants have the necessary knowledge about the screen object.

Direct Manipulation
Franzke [3] also found that participants had trouble on their first encounter with direct manipulation actions like double-click to gain access to an editing dialog box. Kitajima and Polson's [11] action planning model of skilled performance successfully performed such actions. The action planning model had propositions describing links among actions like double-clicking and tasks like edit for each object. Franzke's [3] participants did not have the knowledge that would enable them to infer that an object's attributes could be manipulated by double-clicking on it.

Use of DO-IT Schema
In navigating through the instructions or responding to hints, participants received instructions that described one or more legal actions on a screen object. No participant had any trouble following these instructions. Such instructions are mapped into specific task goals using the DO-IT schema. The resulting task goal is specific enough to enable the action planning process to select the correct object–action pair.

Use of TASK Schema
Participants were asked to "change the legend text to: Geneva, 9, bold." To accomplish this and related tasks in Cricket Graph I, participants had to first double-click on the appropriate text screen object, which opened a dialog box. This dialog box contained a copy of the title-text and three scrolling lists labeled font, size, and style.

Because all participants were experienced Word users, we can assume that they had the knowledge necessary to elaborate this cryptic instruction by assigning appropriate attributes to Geneva, 9, bold, and knowledge to transform them using the TASK schema into a series of subtask goals for font, size, and style. The following is the TASK schema instance for font:

```
TASK schema
        task-action:                      change
        task-object:                      legend-text
        task-specification
                           attribute:     font
                           target:        Geneva
```

Observe that the task instructions do not give any support for finding access to the action double-click.

Franzke [3] found that almost all her participants had to be given a hint to double-click on legend-text. There was no evidence that they had any difficulty forming the correct task goal, change the legend-text. The screen object representing legend-text could easily be identified if participants had general knowledge about graphs. The task goal overlapped with the label for that screen object, so the action planning model would include the correct screen object as one of the

[2] CA Cricket Graph, version 1.3.2, 1989.

[3] CA Cricket Graph III, version 1.01, 1992.

[4] MS EXCEL, version 3.0, 1990.

three candidate screen objects for action. Participants did not know that the legend-text could be double-clicked, or that to edit the legend-text, it must be double-clicked. Thus, there was no link between the action specified by the task goal, change, and the action required by the device to complete this task, double-click. However, the action planning model will never generate the correct action without these links.

Once the dialog box was open, participants had no trouble completing the task. This result is consistent with the LICAI model's behavior. The model can perform each subtask specified by the instruction because the subtask goals link directly to a scrolling list title and to the relevant item in the scrolling list.

Use of DEVICE Schema

Franzke's [3] participants did the graphing task twice. Half did the task again after a 5-min. delay, and the remainder returned in one week. The LICAI model has no learning mechanisms, but participants probably would remember successful task and device goals generated by the various schemata. They would have some chance of remembering hints provided by the experimenter.

There were large practice effects. Mean task completion times dropped from about 15 min. for the first attempt on the task to about 7 min. on the second attempt. There was a small effect of delay of about 1.5 min. Most improvements resulting from practice were found on tasks where terms used in the instructions did not mach labels on the interface (see ref. [3], figures 5, 6, 7, and 8).

Consider the subtask of moving the legend. A significant number of participants had some difficulty with this task during the first session. When this difficulty occurred, the experimenter gave a hint, "Grab the legend which is to the right of the plot symbol, open circle, labeled as Observed." The hint would be comprehended by instantiating DO-IT schema, and the results of performing the hint would be encoded by the following instance of the DEVICE schema:

```
DEVICE schema
  device-object:          Object56
          ID:             legend-text
          attribute
            label:        Observed
            display state: grabbed
  associated-task-goal:   perform "move legend"
```

Object56 represents the legend-text on the graph. Participants would also acquire knowledge that would enable them to correctly recognize new objects as a legend. During the second session, respondents were asked to do the same task 'move the legend,' but with different graph and data. The TASK schema would generate the identical task goal, which would serve as a retrieval cue for device goal. Participants would have acquired knowledge to recognize Object67 as legend and replace Object56 with Object67.

In summary, the large improvements in performance that Franzke [3] observed resulted from improvements on subtasks where the TASK or DO-IT schemata could not generate an effective task goal. One result of successfully performing the interaction is to gain information required by the DEVICE schema to generate the correct device goal. Participants were likely to retrieve these device goals on the second attempt at the task. Kitajima [10] showed that the action planning model always performed correctly when given the correct device goal.

CONCLUSION

We developed and evaluated the LICAI model of display-based human–computer interaction that has goal formation processes and action planning processes, both based on the construction–integration model. The goal formation processes transform initial task descriptions into the precise goals that enable the action planning processes to generate the correct actions. These processes are specialized comprehension strategies that employ task and interface specific schemata to construct goals. The action planning processes use representations strongly constrained by the superficial details of the interface (i.e., labels, menus, and buttons) and the interaction conventions of the host operating system. The goal formation processes must transform task descriptions into goals that link directly to the action planning representations. Most of the power and flexibility of this LICAI model is in the goal formation component.

Implication for Learning by Exploration

Our results have important implications for the development of training materials that support learning by exploration. Carroll [1] summarized an influential research program on the design and evaluation of training materials for application programs. This program led to the development of the Minimalist Instruction paradigm. The minimalist approach focuses on users' tasks rather than on describing a system function by function, minimizes the amount of written material, tries to foster learning by exploration rather than attempting to provide detailed step-by-step instructions, and supports error recognition and recovery.

The development of the Minimalist Instruction paradigm was stimulated by the then-surprising result that detailed and carefully designed training and reference materials for early versions of word-processors were unusable (e.g., [14]). This result is not surprising in light of research on word problems and the LICAI model. Mack et al.'s [14] participants did not have the necessary schemata or action planning knowledge, and attempts to include explicitly all necessary background information lead to long and confusing documentation for these new users.

Our LICAI model suggests that strong constraints on the content of instructional materials exist. Materials generated by minimalist design heuristics are constrained by the interface to an application and users' background knowledge. Although minimalist instructional materials are designed to support learning by exploration, the interface also must facilitate learning by exploration. Otherwise, instructional materials must provide a step-by-step description of how to perform every task. Carroll [1] and his collaborators have shown that most users are unwilling to read such detailed instructions, and users have a great deal of difficulty even if they try to use the step-by-step instructions.

Our LICAI model can be used to develop explicit design guidelines for content. The model's focus on task goals is consistent with the minimalist paradigm. The model makes very clear the kinds of constraints that must be understood in following Carroll's [1] design heuristic of minimizing the amount of written material. We showed that comprehending the very terse instructions used by Franzke [3] required specialized background knowledge about task and interface. "Change the legend text to: Geneva, 9, bold" cannot be understood by someone who has no experience with a modern word-processor. Effectively minimizing the amount of written material requires careful attention to the action and display knowledge and schemata assumed in the user population. A minimalist version of a complete manual for a modern word-processor would have to assume that users have the TASK and DO-IT schemata described in this paper.

The analysis presented in this article strongly reinforces the importance of the label following strategy. In addition, it shows that even when a correct task goal is generated by instantiating TASK and/or DO-IT schemata, a significant amount of background knowledge is needed to select correct actions. These results provide support for the Cognitive Walkthrough [19] methodology, which evaluates the effectiveness of the label following strategy and characterizes the background knowledge necessary to infer correct actions, serving as a design evaluation technique for application interfaces that support learning by exploration.

ACKNOWLEDGMENTS

The authors gratefully acknowledge research support from the National Science Foundation Grant IRI 91-16640. We thank Walter Kintsch and Clayton Lewis for ideas, direction, and comments on earlier versions of this paper.

REFERENCES

1. Carroll, J. M. The Nuremberg Funnel: Designing Minimalist Instruction for Practical Computer Skills. MIT Press, Cambridge, Mass., 1990.
2. Ernst, G.W., and Newell, A. *GPS: A Case Study in Generality and Problem Solving*. Academic Press, New York, 1969.
3. Franzke, M. Turning research into practice: characteristics of display-Based interaction. In *Proceedings of CHI '95*, 421–428. ACM, New York, 1995.
4. Greeno, J.G., and Simon, H.A. Problem solving and reasoning. In R.C. Atkinson, R. Herrnstein, G. Lindzey, and R.D. Luce, eds., *Steven's Handbook of Experimental Psychology*, 589–639. John Wiley and Sons, New York, 1988.
5. Hayes, J. R., and Simon, H.A. Understanding problem instructions. In L.W. Gregg, ed., *Knowledge and Cognition*. Erlbaum, Hillsdale, N.J., 1974.
6. Howes, A., and Young, R. Learning consistent, interactive, and meaningful task-action mappings: a computational model. *Cognitive Science* (in press).
7. Hutchins, E.L., Hollan, J.D., and Norman, D.A. Direct manipulation interfaces. In D.A. Norman, and S.W. Draper, eds., *User Centered System Design*, 87–124. Erlbaum, Hillsdale, N.J., 1986.
8. Kintsch, W. The role of knowledge in discourse comprehension: a construction-integration model. *Psychological Review*, **95**, 163–182, 1988.
9. Kintsch, W., and Greeno, J.G. Understanding and solving word arithmetic problems. *Psychological Review*, **92**, 109–129, 1985.
10. Kitajima, M. Model-based analysis of required knowledge for successful interaction with complex display. ICS Technical Report. Institute of Cognitive Science, University of Colorado, 1995.
11. Kitajima, M., and Polson, P.G. A comprehension-based model of correct performance and errors in skilled, display-based human–computer interaction. *International Journal of Human–Computer Systems*, **43**, 65–99, 1995.
12. Kitajima, M., and Polson, P.G. A comprehension-based model of exploration. ICS Technical Report. Institute of Cognitive Science, University of Colorado, 1995.
13. Larkin, J.H. Display-based problem solving. In D. Klahr and K. Kotovsky, eds., *Complex Information Processing: The Impact of Herbert A. Simon*. 319–342. Erlbaum, Hillsdale, N.J., 1989.
14. Mack, R.L., Lewis, C.H., and Carroll, J.M. Learning to use word processors: problems and prospects. *ACM Transactions on Office Information Systems*, **1**, 254–271, 1983.
15. Mannes, S.M., and Kintsch, W. Routine computing tasks: planning as understanding. *Cognitive Science*, **15**, 305–342, 1991.
16. Payne, S.J., Squibb, H.R., and Howes, A. The nature of device models: the yoked state hypothesis and some experiments with text editors. *Human–Computer Interaction*, **5**, 415–444, 1990.
17. Polson, P.G., and Lewis, C, Theory-based design for easily learned interfaces. *Human–Computer Interaction*, **5**, 191–220, 1990.
18. Polson, P.G., Lewis, C., Rieman, J., and Wharton, C. Cognitive Walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, **36**, 741–773, 1992.
19. Wharton, C., Rieman, J., Lewis, C., and Polson, P. The cognitive walkthrough method: a practitioner's guide. In J. Nielsen and R. Mack, eds., *Usability Inspection Methods*. John Wiley, New York, 1994.