ELSEVIER

Cognitive Science 28 (2004) 901–935

COGNITIVE SCIENCE

http://www.elsevier.com/locate/cogsci

Resolving the paradox of the active user: stable suboptimal performance in interactive tasks

Wai-Tat Fu [a,*], Wayne D. Gray [b]

[a] *Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15232, USA*
[b] *Cognitive Science Department, Rensselaer Polytechnic Institute, Troy, NY, USA*

Received 9 September 2002; received in revised form 2 February 2004; accepted 22 March 2004
Available online 27 July 2004

Abstract

This paper brings the intellectual tools of cognitive science to bear on resolving the "paradox of the active user" [Interfacing Thought: Cognitive Aspects of Human–Computer Interaction, Cambridge, MIT Press, MA, USA]—the persistent use of inefficient procedures in interactive tasks by experienced or even expert users when demonstrably more efficient procedures exist. The goal of this paper is to understand the roots of this paradox by finding regularities in these inefficient procedures. We examine three very different data sets. For each data set, we first satisfy ourselves that the preferred procedures used by some subjects are indeed less efficient than the recommended procedures. We then amass evidence, for each set, and conclude that when a preferred procedure is used instead of a more efficient, recommended procedure, the preferred procedure tends to have two major characteristics: (1) the preferred procedure is a well-practiced, generic procedure that is applicable either within the same task environment in different contexts or across different task environments, and (2) the preferred procedure is composed of interactive components that bring fast, incremental feedback on the external problem states. The support amassed for these characteristics leads to a new understanding of the paradox. In interactive tasks, people are biased towards the use of general procedures that start with interactive actions. These actions require much less cognitive effort as each action results in an immediate change to the external display that, in turn, cues the next action. Unfortunately for the users, the bias to use interactive unit tasks leads to a path that requires more effort in the long run. Our data suggest that interactive behavior is composed of a series of distributed choices; that is, people seldom make a once-and-for-all decision

* Corresponding author.
E-mail address: wfu@cmu.edu (W.-T. Fu).

0364-0213/$ – see front matter
doi:10.1016/j.cogsci.2004.03.005

on procedures. This series of biased selection of interactive unit tasks often leads to a stable suboptimal level of performance.

Keywords: Suboptimal performance; Rationality; Interactive behavior; Cost–benefit tradeoffs; Adaptive choice

Where do inefficient procedures come from and why do people persist in using them? It is a commonplace observation, backed by numerous studies (Bhavnani, 2000; Bhavnani & John, 1996; Carroll & Rosson, 1987; Doane, Pellegrino, & Klatsky, 1990; Nilsen et al., 1993), that people persist in using inefficient procedures for accomplishing a task after hours or even years of experience with a particular artifact (device, tool, or software interface). Carroll and Rosson called this phenomenon the "paradox of the active user." In this paper, we attempt to provide a cognitive science framework to account for this paradox. Although each of our data sets comes from humans interacting with computers, we intend the scope of our questions and framework to be wider than human–computer interaction. We aim at including interactive behavior in any environment in which a human attempts to perform a task using an artifact that has been designed to accomplish that task. Explaining behavior in such task environments requires the weaving together of theoretical mechanisms that are usually discussed in isolation. In the next section, we present the foundation of our approach by first discussing the processes underlying the choice of procedures in interactive tasks. We hypothesize that the display-based nature of interactive tasks leads to a selection bias to procedures, which constitutes the major reason why inefficient procedures persist. We conclude the section by providing our analytic framework. We then apply the framework to analyze each of the three data sets. Finally, we discuss our conclusions and provide our assessment of the adequacy of the framework.

1. The choice of procedures in interactive tasks

Models of adaptive choice such as the Adaptive Control of Thought-Rational (ACT-R, Anderson & Lebiere, 1998), the Adaptive Strategy Choice Model (ASCM, Siegler & Shipley, 1995), and the Adaptive Decision Maker (Payne, Bettman, & Johnson, 1993) assume that past experiences of procedures influence future choice of procedures. The major factors affecting future choice are (1) frequency of use, (2) effectiveness (or accuracy), and (3) efficiency (or speed) of the procedures. Histories on these factors are maintained for each procedure and are updated each time the procedure is used. These histories of past use provide *global data* on the utility of the procedures. In general, procedures that were used more often in the past, or have high effectiveness or efficiency will be more likely to be used in the future. On the other hand, the effectiveness and efficiency of a procedure may differ depending on characteristics of a particular problem. Therefore, the choice of procedures may also depend on *local data*; that is the frequency, effectiveness, and efficiency of each procedure on tasks with a particular set of characteristics. In other words, global data reflect the strength of the general rule that invokes the procedure, whereas local data reflect the strength of the procedure specific to a particular type of problems or contexts[1]. Both ACT-R (e.g., see Taatgen & Anderson, 2002) and ASCM (e.g., see Siegler & Lemaire, 1997) assume that the actual selection process depends on both

global and local data. For example, Siegler and Lemaire showed that when the same problem was frequently encountered, local data dominated over global data in the selection process. On the other hand, when the problem was new or infrequent, global data became dominant.

To illustrate the distinction between global and local data, consider a computer application that allows the user to draw objects and type text on a drawing board. After an object is created, the user can select the object and "drag" it to a desired place on the drawing board by using the mouse. This application also provides a specialized procedure for creating centered text. The specialized procedure is invoked, before typing text, by a command whose immediate result is to open a dialog box. The user can then select the "center" option, close the dialog box, click on the screen at the place about which the text is to be centered, and start to type. As it is typed, the text will flow to the left and right of the centered point automatically. Both the general point-and-drag procedure and the specialized center-text procedure are applicable to creating centered text, but the specialized center-text procedure is more efficient. How do the global and local data of the two procedures influence the selection process when one wants to create centered text in the future? Since the point-and-drag procedure is applicable to any object, its global data is likely to be strengthened frequently. On the other hand, depending on how frequently the specialized centered-text procedure has been used in the past, its predicted effectiveness and efficiency as reflected by its local data is likely to be higher or lower than those of the point-and-drag procedure[2]. In other words, the critical factor that determines which procedure will be chosen is how frequently each procedure has been used to create centered text in the past—if the specialized center-text procedure has been used more often, the local data of the specialized center-text procedure will be strong enough to outweigh the high predicted effectiveness of the general point-and-drag procedure as reflected by its global data. The corollary is that when the specialized procedure is not used often enough in the past, the general procedure may be selected even when the specialized procedure is more efficient.

With all else equal, this reliance on past frequency of use contributes to, but does not seem sufficient to completely explain the paradox of the active user. We propose below that all else is not equal. We will argue that expert users of interactive systems tend to lack deep structural knowledge of these systems and that their behavior tends to be driven by surface characteristics of the display. The result is that even expert users are biased towards interactive actions that lead to incremental change in the external display. This bias tends to reinforce the frequency of general procedure and slow down (or even forestall) the acquisition of more specialized procedures.

1.1. What are the units of decision for interactive behavior?

In traditional problem-solving tasks, expertise is often characterized by deep structural knowledge of the task (e.g., Chi, Feltovich, & Glaser, 1981). We argue that with many or most interactive tasks, the deep structural knowledge is either not so deep or non-existent. For many interactive tasks, behavior is highly dependent on surface features determined by display states (Larkin, 1989). In her seminal paper, Larkin claimed that in many interactive tasks, deciding on the next action is usually trivial because the next task is acquired from information in the external world. For example, when a person brews a cup of coffee using a coffee machine, he or she simply needs to add water if there is no water, add coffee if there is no coffee in the

machine, and so on. The two major characteristics of this example are that control of cognition is mostly display driven, and actions are acquired interactively based on the changes in the external display. This type of display-based interactive behavior has been observed in various tasks in which display states were constantly changing due to on-going interactions between the human and the environment (Kirsh & Maglio, 1994; O'Hara & Payne, 1998). Indeed, expert users of interactive computer applications cannot recall commonly used procedures without the help of the display (Mayes, Draper, McGregor, & Oatley, 1988; Payne, 1991), suggesting a heavy reliance on the display and lack of deep structural knowledge of the systems. Similarly, more recent work (Payne, Richardson, & Howes, 2000) shows that the procedures used in a given display state are much affected by both the familiarity of the display and the frequencies of past successful uses of the procedure.

The display-based nature of interactive behavior does not imply that actions are purely driven by features in the external display. For example, a number of researchers have shown that when subjects solve the eight-puzzle, they often make a few moves to put a piece in a position, pause, make a few moves again, and so on (e.g., Ericsson, 1974, Kotovsky, Hayes, & Simon, 1985; O'Hara & Payne, 1998). Although such behavior is display-based, subjects do not stop and evaluate the display after each move. Rather, a sequence of moves may be executed before subjects pause and evaluate the state of the display. Card, Moran, and Newell (1983) used the term "unit tasks" to characterize this "sequence of short quasi-independent tasks" (p. 385). The scope of our framework is limited to this unit task level of analysis. Once a unit task is selected, executing the steps of a unit task does not require problem solving or deliberate choice. Hence, unit tasks are the smallest units of decision in interactive behavior. This allows us to provide a level of abstraction above the individual-action level while maintaining the power to explain interactive behavior. (A similar argument for the privileged status of the unit task level of analysis has recently been provided by Anderson, 2002). Since each procedure is composed of a set of unit tasks, it is imprecise to imply that a procedure is selected and then executed. Instead, for display-based interactive behavior, a procedure emerges from the successive selection of unit tasks where each prior unit task affects the external display, which in turns cues for the next action. As we discuss in the next section, the series of distributed, display-based choices of unit tasks is a characteristic of interactive tasks that induces the persistent of inefficient procedures.

2. Acquisition and persistence of inefficient procedures

We propose that inefficient procedures are composed of unit tasks that are either used in similar task environments, or used elsewhere within the current task environment. From earlier discussions, global data (past experience of procedures in all environments) are weighed more heavily than local data in the new environment simply because there is a lack of specific experience with the unit tasks in the new environment. For example, in word processing, a general procedure might lead the user to indent a paragraph by using the space bar or tab key rather than a specialized style sheet. The implicit assumption of the developer is that, as the user gains experience, these general procedures will be supplanted by more specialized and efficient procedures. However, as we can anecdotally judge from numerous electronic

submissions of student papers, such general procedures persist even after years of experience with word processors.

We argue that it is often the case that specialized procedures are the most efficient procedure only under specific conditions. In our last example, using the spacebar or tab key may be more efficient than using the specialized style sheet when there is only one paragraph to be indented. This implies that evoking the most efficient procedures often requires high frequency of use under the same conditions.

We hypothesize that the selection of unit tasks may be biased[3] towards those that are composed of interactive actions. Each of these interactive actions leads to incremental change in the external display. In turn, this change cues the next action towards the goal. One possible explanation for the bias is that these interactive unit tasks require less cognitive effort as mental look-ahead can be off-loaded to the external display. In addition, incremental changes allow better error control as immediate feedback on actions is available. We argue that the bias in unit task selection may lead to selection of general, but often inefficient procedures. For example, when the initial unit task of procedure A is more interactive than that of procedure B, the initial unit task of procedure A is more likely to be selected. This may eventually lead to the use of procedure A, even if the overall efficiency of procedure B is better.

2.1. Why do inefficient procedures persist?

As predicted by adaptive choice models, the more a procedure was used in the past, the more likely it will be used in the future. The more that specialized procedures are used, the more likely it is that their local data will outweigh the global data of general procedures, leading to an eventual supplanting of the general procedures. However, we hypothesize that this process is much slower in interactive tasks than non-interactive tasks in which there is less reliance on the external display. The bias in selecting interactive unit tasks limits the use of the efficient procedures. This bias results in a less than expected use (and less strengthening) of specialized procedures, and a greater than expected use (and further strengthening) of general procedures composed of interactive unit tasks.

2.2. Support for framework: three questions

To support our framework we look for evidence in three data sets of interactive behavior. For each data set we look for cases where people are not using the recommended procedures provided by their artifacts. For these cases, we ask three questions. First, are the recommended procedures more efficient than the procedures that people actually used (we call these the preferred procedures); that is, can we document Carroll and Rosson's (1987) paradox of the active user? Second, is it the case that the recommended procedures really are either more specific to the program or more specialized within the program than the corresponding preferred procedures? Third, are the preferred procedures composed of components that are more interactive than the recommended procedures? Specifically, we analyze the initial unit tasks of the preferred and recommended procedures and ask which procedure is able to provide faster incremental feedback to the user as reflected by changes in the external display.

We ask these three questions of data obtained from three widely varying sets of users and software. The first data set was collected from Education graduate students after they had spent approximately 20 and 40 h learning a simple end-user, multimedia authoring system, HyperCard™. Our subjects' inexperience allows us to assess how inefficient procedures are acquired. Our second data set is limited to two examples of an expert architect using an architectural CAD system (Bhavnani, 2000; Bhavnani & John, 1996, 1997, 1998, 2000; Bhavnani, John, & Flemming, 1999). For each example, we compare Bhavnani and John's analysis of the procedure used by the expert with their analysis of the optimal procedure. For each case, we apply our framework to ask the same three questions as we do of the first data set. This data set allows us to study how inefficient procedures persist even after years of practice. For our third study, we conducted a controlled experiment to directly test our hypotheses that selection is biased toward general, interactive procedures.

3. Study I: natural learning in an artificial task environment

The first set of observations centered on natural learning in a classroom. The participants were School of Education graduate students enrolled in a course to learn HyperCard™. HyperCard provides an interactive, direct manipulation interface for the users to create their own objects. We asked the students to complete two benchmark tasks during the course. All of the skills and knowledge required by the first benchmark task were required also by the second benchmark task. In addition, the second benchmark task required skills and knowledge not needed by the first benchmark task.

3.1. Subjects

Nine Education graduate students enrolled in a HyperCard course (using HyperCard 2.0) participated in the study. The students were learning how to use HyperCard to build educational tools. At the beginning of the course, students were asked to self-report their experience with computers. None of the students had experience in computer programming, all of them had some experience in using simple word-processing software, and few had any experience with drawing programs. They were all familiar with basic computer operations like pointing and clicking, pulling down menus, and choosing menu items. Students followed their textbook (Beekman, 1991) and worked through all the examples and assignments during the course. Students were required to keep weekly records of the number of hours spent on each aspect of coursework.

3.2. The tasks

At the time of the first benchmark task (BM1), the students had an average of 22 h of HyperCard experience. At the time of the second benchmark task (BM2), they reported a mean of 46 h of HyperCard experience. As students completed the benchmark tasks, digital movies were made. Unfortunately, one digital movie in BM1 and three digital movies in BM2 were incomplete, but the final stacks[4] of two of the BM2 movies were intact. This attrition left us

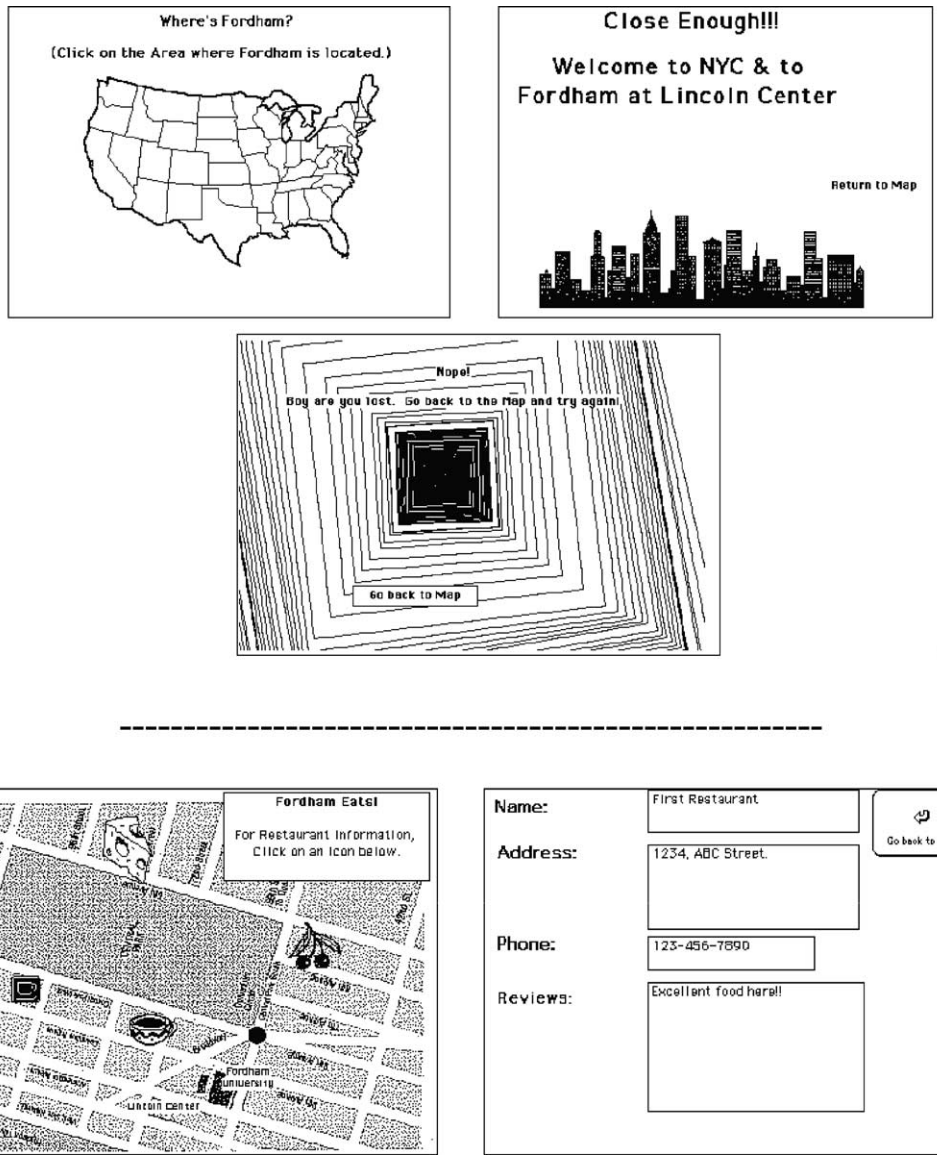

Fig. 1. Screen prints of the three cards built by the students in the first benchmark task (BM1) (Top), and screen prints of the street map card and the restaurant information card in the second benchmark task (BM2) (Bottom). Five restaurant information cards were required, but only the contents were different.

with eight movies in BM1 and six movies in BM2 (therefore only six students for whom we had protocols for both BM1 and BM2 tasks), and eight finished stacks in both BM1 and BM2. In both tasks, students were given the instructions and the sample screen prints of the completed stacks (i.e., the goal state, see Fig. 1). Variations of the positions of the buttons, text style and font, and the user-created graphics were acceptable. The instructor answered all questions the

students had and stayed until the students had finished their tasks. Students finished their tasks at their own pace.

3.3. The first benchmark task

The first benchmark task, BM1, was designed to test the basic knowledge the students learned in the first 4 weeks of the course. They had to build a three-card HyperCard stack. There were four basic objects to create: cards, text, graphics, and linked buttons. Screen prints of each of the cards were given to them (top of Fig. 1). The task took the students an average of 30 min to complete. At the time they did BM1, all the basic tasks had been accomplished successfully at least twice.

3.4. The second benchmark task

The second benchmark task, BM2, was designed so that performance and learning could be compared to BM1. Similar to BM1, students were required to build buttons, type text, link buttons to cards, and copy and paste graphics. They had to create one street map card and five restaurant information cards (bottom of Fig. 1). Only the contents in the five restaurant information cards were different. The task took the students an average of 45 min to complete.

Students finished Chapter 4 before they did BM2, in which they learned how to use the background layer in HyperCard. The background layer provides a primitive form of inheritance. Anything on the background layer (graphics, text, field boxes, or buttons and their actions) is contained on every new card the user creates. While creating a stack, users can switch back and forth between the background layer and the card layer. They were also taught to use field boxes. A field box contains editable text. It can be resized and moved around on screen with the text it contains. Attributes of the text inside a field box can be changed anytime. (The field box text is ASCII and has most of the same properties as in a word processor. The text used in BM1 was bitmapped and once it was created, it could only be erased and replaced, not edited.) Both the background layer and the field box allowed the students to work more efficiently in BM2 than in BM1.

3.4.1. Transcription and task analysis

One of the goals of our analyses is to measure efficiency of procedures by their execution times, so that we can compare the efficiency of the procedures recommended by the textbook and the non-recommended procedures used by the students. However, since we did not collect our data in a controlled experiment, students were often distracted, and they often made mistakes or slips while they were working on the tasks. Besides, students only did their tasks once, thus we could only directly measure the execution times for the procedures used by the students (either recommended or non-recommended, but not both). We chose to solve this problem by deriving engineering models of the procedures and estimate the execution times based on the models instead of making direct measurements. Engineering models derive mean execution times for operators based on psychological literature, thus served our purpose of comparing efficiency of procedures and at the same time controlling for individual differences in execution times for different operators.

All movies were transcribed and the action protocols were coded using seven operators: Mouse Down, Mouse Up, Click, Double Click, Mouse Down and Drag, Move Cursor, and Type. This set captures the basic action sequences of the students in both benchmark tasks. To capture the higher-level operators, GOMS models were derived. GOMS success in such situations is well documented (see, e.g., Gray, John, & Atwood, 1993). GOMS, Goals, Operators, Methods, and Selection Rules, is a technique for cognitive task analysis (Card et al., 1983). We began by creating GOMS analyses of the procedures recommended by the textbook. These analyses were based on the step-by-step instructions recommended by the textbook. The recommended procedures were the most efficient ones known to the students. Since the current focus was on how the students deviated from the procedures that they had been taught, preferred procedures for the same tasks were also characterized using GOMS to allow easy comparisons.

3.4.2. Results

The GOMS analyses of recommended procedures were used to trace (see also, Anderson, Boyle, Corbett, & Lewis, 1990; Gray, 2000; Gray & Anderson, 1987) the action protocols generated by the students. The use of each of the recommended procedures was counted. Episodes involving slips or mistakes that were detected and corrected before the student attempted another goal were collapsed; that is, the entire episode was considered as one of the correct recommended procedures. Whenever the students used or attempted to use a procedure not taught by the textbook, we called this a *preferred procedure* and performed a GOMS analysis on it. Preferred procedures were found to be pervasive throughout both sets of protocols. The recommended and preferred procedures together captured 95% of the action protocols. The remaining protocols appeared to be more random than systematic, and have eluded our analyses.

3.5. Number of procedures used

In counting the number of procedures used, we distinguish between types and tokens. For example, a student who only used the most efficient procedures taught by the textbook (an ideal student) would use 15 different types of procedures in BM1 and 17 different types of procedures in BM2. Each use of a procedure represents a different token of that type of procedure. As most procedure types are used multiple times, our ideal student would use a total of 79–120 procedure tokens in BM1 and from 173 to 227 procedure tokens in BM2. The range in number of tokens per benchmark task is possible as different recommended procedures exist for accomplishing the same goal and these procedures are composed of varying numbers of other procedures.

As Table 1 shows, by the low scores on standard deviation, the number of types of recommended procedures used was constant. With one exception, all students used 15 types of recommended procedures in BM1 and all used 17 types of recommended procedures in BM2. For BM1, one student used two additional recommended procedures. These procedures accomplished the task she/he was trying to achieve but were less efficient than the recommended procedures used by the other students. For the eight sets of protocols in BM1 the number of tokens of recommended procedures used by the students ranged from 77 to 95 (mean = 87, see Table 1), whereas for the 6 sets of protocols in BM2 the number of recommended procedures ranged from 164 to 197 (mean = 184).

Table 1
Number of recommended and preferred procedures used by the students in the benchmark tasks

	Recommended procedures			Preferred procedures		
	Types	Tokens	*n*	Types	Tokens	*n*
Benchmark 1						
M	15	87	8	3	11	8
SD	0.71	7.48		1.83	6.53	
Benchmark 2						
M	17	184	6	3	10	6
SD	0.00	11.58		1.10	4.05	

M = Mean; *SD* = Standard deviation.

Although all students successfully accomplished the two benchmark tasks, none limited themselves to recommended procedures. Table 1 also shows the number of types and tokens of preferred procedures used. For BM1 the number of types of preferred procedures per student ranged from one to six (mean = 3), whereas the number of tokens ranged from 4 to 22 (mean = 11). For BM2 the types of preferred procedures per student ranged from two to five (mean = 3), whereas the tokens ranged from 4 to 14 (mean = 10).

Chapter 4 of the textbook introduced new procedures to cover new tasks, but continued to use the older procedures as appropriate. Hence, it is not surprising that the already high use of the BM1 recommended procedures increased in BM2. However, despite mastering most types of recommended procedures, Table 1 indicates that students continued to deploy many tokens of alternative procedures. In the following sections, we will turn our attention to the analysis of these alternative procedures.

3.6. Descriptions of the preferred procedures

Table 2 lists the preferred procedures used in BM1 and BM2. For the successful procedures, these attempts to invent new procedures worked. For the unsuccessful procedures, the attempt failed. In this section, we discuss two of the successful preferred procedures shown in Table 2.[5]

Our task analysis uses the NGOMSL (Kieras, 1997) dialect of GOMS. In deriving time estimates for the procedures, we rely on the default times provided by NGOMSL. These default times were originally derived from estimates in the psychological literature (e.g., see Card et al., 1983). Use of default times provides a control for individual differences which, given our small sample sizes, might otherwise distort our comparisons.

3.6.1. Center title

To center the title of a card, students using the recommended procedure first opened the text-tool palette, selected the attribute “centered,” closed the palette, clicked on the center of the card, and began typing. In contrast, students using the preferred procedure first typed the title on the card, then clicked on the lasso[6] tool, used the lasso to select the text, mouse downed on the selected text and dragged it to the desired location. Both the recommended procedure

Table 2
Types and tokens of preferred procedures for benchmark tasks 1 and 2

	Benchmark 1	Benchmark 2
Successful		
Use lasso to center text[a]	8	4
Type button name on card[a]	4	4
Create new button just to change mode[b]	5	0
"Deleted" button by covering it with a white patch[b]	1	0
Use spacebar to center text in field box[b]	NA	1
Create buttons by copy-paste[b]	0	1
Copy-paste-edit text instead of direct typing[b]	NA	4
Unsuccessful		
Draw button[b]	6	0
Use lasso to select button or fields[b]	1	2
Use eraser to delete button[b]	1	0
"Deleted" button by covering it with a black patch[b]	1	0
Use spacebar to center button name[b]	2	0
Attempt to type bitmapped text into ASCII field box[b]	NA	6
Attempt to type ASCII text into bitmapped rectangle[b]	NA	1
Test link on a not yet built transparent button[c]	0	1

The types are specified in the second column, tokens indicate number of times each procedure was used. Students might have used the procedures more than once, see text for details. NA: preferred procedures that were used for Benchmark 2 that could not have been applied to Benchmark 1.

[a] Example discussed in text.

[b] Example discussed in the appendix, which can be found in http://cogsci.psy.utexas.edu/supplements/.

[c] Example not provided.

(Table 3) and preferred procedure (Table 4) for center-title consists of two unit tasks one of which, type-text, is common to both. The other unit task differs between procedures but, in each case, is one that occurs in a recommended procedure. Hence, the preferred procedure is a non-recommended combination of two recommended unit tasks.

As shown at the bottom of each table, the NGOMSL analysis predicts a total time for recommended procedure of 13.1 and 16.8 s for the preferred procedure. In answer to our first question, the analysis supports the assertion that the recommended procedure is more efficient than the preferred procedure. As to which procedure is the more specialized, the NGOMSL analysis shows that the two have one unit task in common and differ on whether their other unit task is move-bitmap-to-<location> (for the preferred procedure) or change-<object>-attribute (recommended procedure). Of these two, the students would have had much practice with using the lasso tool to move bitmapped graphics or bitmapped text from one card location to another, using the procedure shown in Table 4. In contrast, although change-<object>-attribute(s) is used frequently within HyperCard, each object–attribute combination entails a different procedure. On the other hand, the move-bitmap-to-<location> has the exact same procedure for bitmaps, buttons and other HyperCard objects. In fact, the click-and-drag procedure is common to most direct manipulation interfaces. By this analysis, the recommended procedure is specialized and the preferred procedure is general.

Table 3
GOMS analysis showing the recommended procedure for the unit task, center-title

TRACE of NGOMSL model for the recommended procedure of center-title	Time (s)
AG: create-title	0.1
AG: change-<object>-attribute(s) (set button to centered) [[unit task]]	0.1
AG: activate-<object>-palette (for text mode)	0.1
DECIDE: IF hand not on mouse then OPERATOR: HOME hand to <mouse>	0.1
OPERATOR: MOVE cursor to <text-button>	1.1
OPERATOR: DOUBLE-CLICKON <text-button>	0.4
RETURN with goal accomplished	0.1
AG: set-<attribute> (center)	0.1
OPERATOR: MOVE cursor to <align-center>	1.1
OPERATOR: CLICKON <align-center>	0.2
RETURN with goal accomplished	0.1
AG: close-palette	0.1
DECIDE: IF <new-settings-ok> THEN OPERATOR: MOVE cursor to <ok> ELSE OPERATOR: MOVE cursor to <cancel>	1.2
OPERATOR: CLICKON-<button>	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
Unit task time	5.1
AG: type-text [[unit task]]	0.1
DECIDE IF text mode not selected THEN AG: select-text-mode	0.1
AG: position-<insert-point>-on-card	0.1
OPERATOR: MOVE cursor to <card-loc>	1.1
OPERATOR: CLICKON <card-loc>	0.2
RETURN with goal accomplished	0.1
OPERATOR: HOME hand to <keyboard>	0.4
OPERATOR: TYPE-<string> (assume 20 letters)	5.6
RETURN with goal accomplished	0.1
Unit task time	7.8
RETURN with goal accomplished	0.1
Total time	13.1

Center-title consists of two unit tasks, create-title and create-text-on-screen both of which follow recommended procedures. AG: accomplish goal.

At the point at which students decide to create a title, our analysis suggests that there are two unit tasks competing as the first step: type-text (for the preferred procedure) versus change-<object>-attribute (recommended procedure). After each action in the unit task for the more general procedure (approximately every 0.3 s after typing each letter), the system provides immediate feedback on the incremental change on the problem state. On the other hand, change-<object>-attribute takes 5.1 s and even after this unit task, no change to the display can be observed until the title is typed. We therefore conclude that, in addition to being more general, the initial components of the preferred procedure are more interactive than those of the more specialized, recommended procedure.

Table 4
GOMS analysis showing the preferred procedure for the unit task, Center Title

TRACE of NGOMSL model for the preferred procedure of center-title	Time (s)
AG: create-title	0.1
AG: type-text [[unit task]]	0.1
DECIDE IF text mode not selected THEN AG: select-text-mode (we assume here that it is not already selected)	0.1
AG: select-mode	0.1
OPERATOR: MOVE to <text tool in Tool Palette>	1.1
OPERATOR: CLICKON <text tool>	0.2
RETURN with goal accomplished	0.1
AG: position-<insert-point>-on-card	0.1
OPERATOR: MOVE cursor to <card-loc>	1.1
OPERATOR: CLICKON <card-loc>	0.2
RETURN with goal accomplished	0.1
OPERATOR: HOME hand to <keyboard>	0.4
OPERATOR: TYPE-<string> (assume 20 letters)	5.6
RETURN with goal accomplished	0.1
Unit task time	9.3
AG: move-bitmap-to-<location> [[unit task]]	0.1
AG: select-<lasso-mode>	0.1
DECIDE: IF hand not on mouse then OPERATOR: HOME hand to <mouse>	0.5
OPERATOR: MOVE cursor to <lasso tool in Tool Palette>	1.1
OPERATOR: CLICKON <lasso tool>	0.1
RETURN with goal accomplished	0.1
AG: select-<title-text>-using-lasso	0.1
OPERATOR: MOVE cursor to <near text>	1.1
OPERATOR: MOUSEDOWN and circle <text>	1.2
OPERATOR: MOUSEUP	0.1
RETURN with goal accomplished	0.1
AG: move-<object>-to-<location>	0.1
OPERATOR: MOVE cursor to <bitmap>	1.1
OPERATOR: MOUSEDOWN	0.1
OPERATOR: DRAG selection	1.1
OPERATOR: MOUSEUP	0.1
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
Unit task time	7.3
RETURN with goal accomplished	0.1
Total time	16.8

The invention is a non-recommended combination of two unit tasks that incorporate recommended procedures (create-text-on-screen and move-<text>-to-<center-of-line>). AG: accomplish goal.

Table 5
GOMS analysis showing the recommended procedure for the unit task, create-button-label

TRACE of NGOMSL model for the recommended procedure of create-button-label	Time (s)
AG: change-<object>-attribute(s) [[unit task]]	0.1
AG: activate-<object>-palette	0.1
DECIDE: IF hand not on mouse then OPERATOR: HOME hand to <mouse>	0.1
DECIDE: IF button-mode NOT selected THEN AG: select-button-mode	0.1
OPERATOR: MOVE to <button>	1.1
OPERATOR: DOUBLE-CLICKON <button>	0.4
RETURN with goal accomplished	0.1
AG: set-<attribute> (label)	0.1
OPERATOR: MOVE cursor to <label-field>	1.1
OPERATOR: CLICKON <label-field>	0.2
OPERATOR: HOME hand to <keyboard>	0.4
OPERATOR: TYPE-<string> (assume 10 letters)	2.8
RETURN with goal accomplished	0.1
DECIDE: IF <show button name> NOT selected THEN AG: set-button-attribute-<show-button-name> (As "show button name" is the default setting, we assume it is set.)	0.1
AG: close-palette	0.1
DECIDE: IF <new-settings-ok> THEN OPERATOR: MOVE cursor to <ok> ELSE OPERATOR: MOVE cursor to <cancel>	1.2
OPERATOR: CLICKON-<button>	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
Unit task time	8.5

The analysis assumes that the student has just created the button and that HyperCard is in "button mode." AG: accomplish goal.

Three of the nine students used the preferred procedure. Their usage of the preferred procedure cannot be attributed to ignorance of the existence of the text palette. Each of the students had used the text palette several times to select other text attributes (such as *bold*). These actions demonstrated that these students knew about the existence of the palette, how to evoke it, and its relevance to text. However, rather than using the palette to center text, the students followed the procedure shown in Table 4. In BM2, one of the three students who had used the preferred procedure in BM1 switched to using the recommended procedure to center titles, but the other two continued to use the more general and interactive procedure.

3.6.2. Create button label

As shown in Table 5, we have modeled the recommended procedure for create-button-label as consisting of a single unit task, change-<object>-attribute(s). This unit task is the same as the first unit task in the procedure for center-title (as discussed immediately above), though the procedure is different. The preferred procedure works by typing the label directly on the card, changing the button attribute to "transparent," and then moving the button to on top of its label. This procedure consists of three unit tasks: type-text, move-<object>-to-<location>, and change-<object>-attribute (see Table 6).

Table 6
GOMS analysis showing the preferred procedure for the unit task, create-button-label

TRACE of NGOMSL model for the preferred procedure of create-button-label	Time (s)
AG: create-button-label	0.1
AG: type-text [[unit task]]	0.1
DECIDE IF text mode not selected THEN AG: select-text-mode (we assume here that it is not already selected)	0.1
AG: set-<mode>	0.1
OPERATOR: MOVE cursor to <text tool in Tool Palette>	1.1
OPERATOR: CLICKON <text tool>	0.2
RETURN with goal accomplished	0.1
AG: position-<insert-point>-on-card	0.1
OPERATOR: MOVE cursor to <card-loc>	1.1
OPERATOR: CLICKON <card-loc>	0.2
RETURN with goal accomplished	0.1
OPERATOR: HOME hand to <keyboard>	0.4
OPERATOR: TYPE-<string> (assume 10 letters)	2.8
RETURN with goal accomplished	0.1
Unit task time	6.5
AG: move-<object>-to-<location>	0.1
OPERATOR: MOVE cursor to <button>	1.1
OPERATOR: MOUSEDOWN	0.1
OPERATOR: DRAG selection	1.1
OPERATOR: MOUSEUP	0.1
RETURN with goal accomplished	0.1
Unit task time	2.6
AG: change-<object>-attribute(s) [[unit task]]	0.1
AG: activate-<object>-palette	0.1
DECIDE: IF hand not on mouse then OPERATOR: HOME hand to <mouse>	0.5
OPERATOR: MOVE cursor to <button>	1.1
OPERATOR: DOUBLE-CLICKON <button>	0.4
RETURN with goal accomplished	0.1
AG: set-<attribute> (transparent)	0.1
OPERATOR: MOVE cursor to <transparent> radio button	1.1
OPERATOR: CLICKON <transparent>	0.2
RETURN with goal accomplished	0.1
AG: set-<attribute> (no-name)	0.1
OPERATOR: MOVE cursor to <show-name>	1.1
OPERATOR: CLICKON <show-name> (this “deselects” default option)	0.2
RETURN with goal accomplished	0.1
AG: close-palette	0.1
DECIDE: IF <new-settings-ok> THEN OPERATOR: MOVE cursor to <ok> ELSE OPERATOR: MOVE cursor to <cancel>	1.2
OPERATOR: CLICKON-<button>	0.2
RETURN with goal accomplished	0.1
Unit task time	6.9
RETURN with goal accomplished	0.1
Total time	16.2

AG: accomplish goal.

As shown at the bottom of each table, the NGOMSL analysis predicts a total time for recommended procedure of 8.5 s, and a predicted time of 16.2 s for the preferred procedure. In answer to our first question, the analysis supports the assertion that the recommended procedure is more efficient than the preferred. As to which procedure includes the more specialized unit tasks, the NGOMSL analysis shows that the preferred procedure contains the same specialized unit task as the recommended procedure (change-<object>-attribute), but its first two unit tasks are very general (type-text and move-<object>-to-<location>). This is an interesting case where the more general, interactive unit tasks lead the subject down the path where the only thing that works is the slower, more specialized, change-<object>-attribute, unit task.

It should be noted that the procedure shown in Table 6 is the product of problem solving. Initially, the students who used this procedure attempted to type directly on the button object—not on the card. However, HyperCard does not allow users to type anything on the button. When students attempted to type on the button, the text was typed on the card beneath the button. Hence, the button label was covered by the button and could not be seen. The students had to drag away the button to know that the button label was typed correctly. They then opened the button palette, clicked on the checkbox labeled "transparent," and unchecked the checkbox labeled "show button name." Finally, they dragged the button back over the label.

Similar to the center-title unit task, the type-text unit task in the preferred procedure leads to immediate feedback as reflected in the external problem state. On the other hand, the recommended procedure used the change-<object>-attribute unit task that took 5.1 s before the problem state was changed. We therefore conclude that the initial unit task of the preferred procedure was more interactive than that of the recommended procedure.

Surprisingly, from BM1 to BM2 the number of students who used the preferred procedure increased. Although the BM2 protocols of the two students who used the preferred procedure in BM1 were incomplete, by examining the final stacks they built, we found that they (S7 and S8) continued to use the preferred procedure. Contrary to the expectation that people adopt more specialized procedures with experience, S4 and S6, who did not use the preferred procedure to create button labels in BM1, used the procedure in BM2. What is even more interesting is that two of the students, S4 and S7, had modified the preferred procedure to increase the level of support from the external display. Instead of typing the button label on the button, they typed the button label next to the button on the card. In this way, they could make sure that the label was typed correctly. They dragged the button back over the label and then made it transparent (this is the procedure that we model in Table 6). These two students had invented a new procedure based on the apparently more interactive initial unit task.

3.7. Summary of observations in the HyperCard task: answers to the three questions

How does our analysis of the procedures used in HyperCard address our three questions? First, as judged by predicted total time, the recommended procedures are more efficient than the corresponding preferred procedures. Hence, we have documented *the paradox of the active user*. Second, the specialized unit task, change-<object>-attribute, in the recommended procedures tend to be more specific to HyperCard and more specialized within HyperCard than the corresponding unit tasks, type-text and move-<object>-to-location, of the preferred procedures. In fact, both unit tasks in the preferred procedures are common procedures found in

most interactive interfaces with which the students reported they had experienced. The picture that emerges is that if specialized procedures were well-practiced they would have been used to accomplish the task more efficiently than the preferred procedures. However, the preferred procedures contain general, interactive unit tasks that would have been practiced many places within HyperCard. The use of these preferred procedures is therefore consistent with our hypothesis that (1) there is a bias towards interactive unit tasks and this bias may lead to the use of inefficient procedures, or worse still, problem-solving behavior and invention of new, albeit inefficient procedure; and (2) when the frequency of use of the specialized procedure is low compared to the general procedure, the local data for the specialized procedure is not strong enough to completely dominate over the global data of the general procedures. The analyses also show that selection of procedures is composed of a distributed series of choices of unit tasks. At the point where different procedures were to be selected, an initial unit task that was more general and interactive was often selected, which eventually led to a series of unit tasks that constituted an inefficient procedure.

4. Study II: the behavior of professional architects in the CAD system

4.1. The use of CAD system for architectural designs

The analyses of the HyperCard students support our hypothesis that during the early stages of learning, inefficient procedures are acquired (or invented) based on general procedures from either the same environment (i.e., HyperCard) or a wide range of similar environments (i.e., other computer applications such as word processors or spreadsheet software). To test our hypothesis that these inefficient procedures persist even after years of experience, we reanalyzed two sets of data collected by Bhavnani and John (1996, 1997, 1998, 2000) and Bhavnani et al. (1999) from real-world CAD usage by architects. They showed that even after years of experience, architects tend to use inefficient procedures to perform complex CAD tasks. The architects were different from the novice users in the HyperCard system as they possessed sophisticated general (how architectural objects are designed) and specific (how architectural objects are created in manual drafting environments) task knowledge, and had used the architectural CAD system, daily, for several years.

4.2. Task 1—drawing complex shapes with lines and arcs

The task was to draw three identical complex shapes consisting of lines and arcs (Fig. 2). In the CAD system, the efficient, recommended procedure is to draw the arc and all the lines of the shape, group them, and then make two copies (the Detail–Aggregate–Manipulate, see Fig. 2). In contrast, for manual drafting an efficient procedure would be to draw the three arcs first (minimizing changes to the setting of the compass), then draw the vertical and horizontal lines for each arc. Although this preferred procedure is efficient for manual drafting, it is less efficient than the recommended procedure for CAD drafting. Despite its inefficiency, Bhavnani and John observe that the preferred procedure is commonly used.

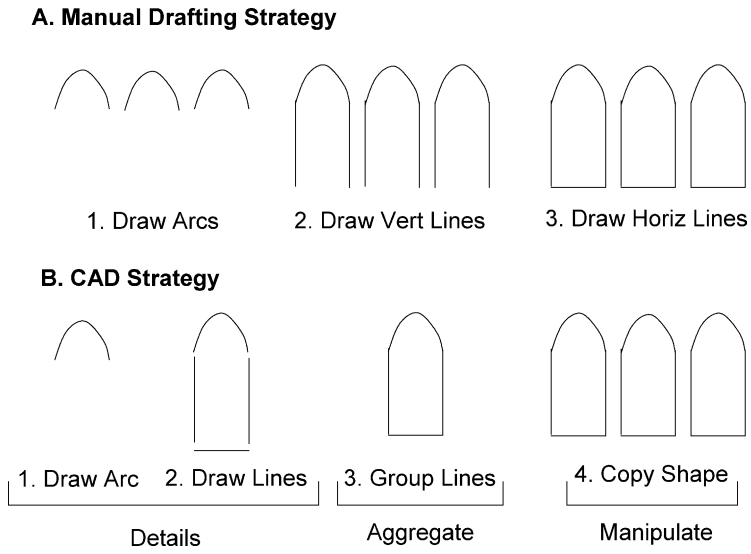

Fig. 2. Example of drawing three arcs (adapted from Bhavnani & John, 1996).

Although we do not have an NGOMSL analysis of this task, we agree with Bhavnani and John's that the recommended procedure is clearly more efficient than the preferred procedure; hence, in answer to our first question, there is a paradox that needs to be explained. For our second question, apparently the inefficient procedure is more general than the recommended procedure. As the analyses by Bhavnani and John showed, the specialized procedure is more efficient only when the number of complex shapes is equal to or more than three. It seems plausible that when they started to learn the software they might have started with a single simple shape. It is not clear to us how often architects are called on to draw complex versus simple shapes. However, even if the ratio of complex to simple drawings is high, the frequency of use of the specialized procedure might not be enough that the local data would cause the specialized procedure to completely dominate over the general procedure that is applicable to any number of complex or simple shapes. Indeed, Bhavnani and John found the even after years of experience of the applications, these inefficient procedures were seldom completely weeded out.

Third, both procedures begin by selecting the arc-tool, moving to the correct location, and drawing one arc. The general procedure continues by drawing two additional arcs. In contrast, drawing the next object using the specialized procedure requires the architect to move the cursor to the tool palette, select the line tool, move the cursor back to the first arc, and then draw a line. Although we do not have a GOMS analysis for this task, but we find this procedure similar to the recommended procedures for Center Title and Create-Button-Label in Study I. From previous analyses, using the tool palette takes more time to create a change on the display than directly drawing the next object. We therefore conclude that the initial components of the preferred procedure of draw-next-object are more interactive than that for the recommended procedure of get-new-tool-&-draw-line-connected-to-arc. Drawing another arc directly brings the problem state closer to the desired state and is therefore more interactive than the recommended procedure.

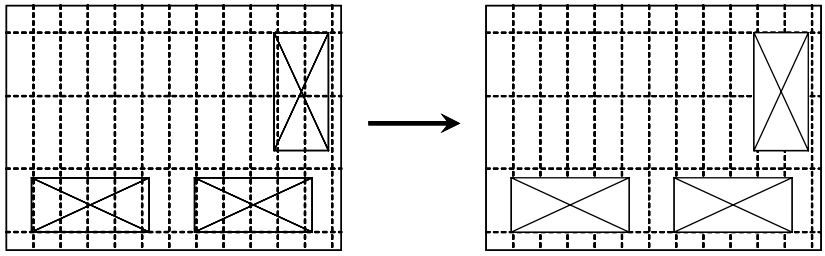

Fig. 3. The panel clean-up task requires all ceiling panel lines that overlap the air-condition vents to be modified. The drawings are schematic and not to scale (Bhavnani & John, 1998).

4.3. Task 2—the panel clean-up task

The panel clean-up task (Bhavnani & John, 1998, 2000) entails editing an architectural plan containing ceiling panels (the line segments) that overlapped air-conditioning vents (the rectangles) (Fig. 3). The architect was to remove all line segments that overlapped the rectangles. An expert architect who had 2 years of daily experience with the CAD system used an inefficient procedure. He first cut all panel lines that overlapped the vents (see Fig. 4) by using the "element" tool, which deletes a portion of a given line between two specified points. He then cleaned up all the cut-lines to the edges of the vent using the "intersection" tool, which extends or shortens a line to the intersection point of any other line. Bhavnani and John contrast this inefficient procedure with a more efficient and specialized procedure in which the entire vent is selected with an aggregate tool and then all panel lines are deleted in one step. The specialized procedure can then be applied to the other vents (the aggregate-drop-modify strategy). The GOMS models for the two procedures are shown in Tables 7 and 8.

According to Bhavnani and John (2000), "Ethnographic notes revealed that [the expert] had used the place fence command several times in other tasks to modify groups of objects. The missed opportunity to use [that strategy again] was therefore not due to the lack of knowledge of this command" (footnote 1, p. 118). Indeed, Bhavnani and John have identified a paradox as, in answer to our first question, NGOMSL analyses[7] suggest that the preferred procedure would take three times as long (140.3 s) for the three-vent task than would the recommended procedure (42.4 s).

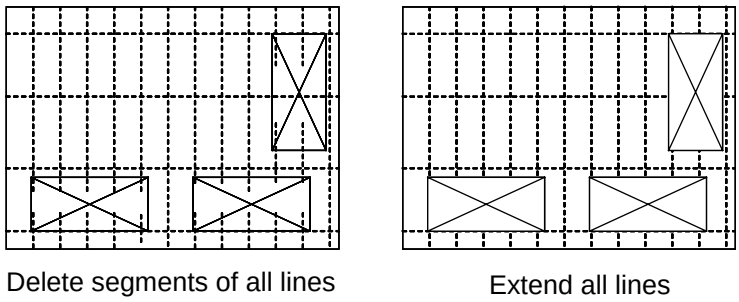

Fig. 4. The method used by Bhavnani and John's expert to perform the panel clean-up task (Bhavnani & John, 1998).

Table 7
The trace of the NGOMSL model of the recommended procedure for the panel clean-up task discussed in Bhavnani and John (2000)

TRACE of NGOMSL model for the recommended procedure of clean-up vent	Time (s)
AG: cleanup-vent	0.1
AG: aggregate [[unit task]]	0.1
AG: setup-command-"Place-Fence"	0.1
OPERATOR: MOVE to <cmd1>	1.1
OPERATOR: CLICKON button	0.2
OPERATOR: MOVE to <cmd2>	1.1
OPERATOR: CLICKON button	0.2
OPERATOR: MOVE to <cmd3>	1.1
OPERATOR: CLICKON button	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
AG: place-fence	0.1
OPERATOR: LOCATE-VERTEX-1	0.23
DECIDE: PRECISE-POINT OR NOT?	0.1
OPERATOR: MOVE to vertex-1	1.1
OPERATOR: CLICK	0.2
VERIFY snap	0.15
OPERATOR: CLICK	0.2
OPERATOR: LOCATE-VERTEX-2	1.1
DECIDE: PRECISE-POINT OR NOT?	0.1
OPERATOR: MOVE to vertex-2	1.1
OPERATOR: CLICK	0.2
VERIFY snap	0.15
OPERATOR: CLICK	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
Unit task time	9.33
AG: modify [[unit task]]	0.1
AG: setup-command-"Clip-and-Delete"	0.1
OPERATOR: MOVE to cmd1	1.1
OPERATOR: CLICKON cmd1	0.2
OPERATOR: MOVE to cmd2	1.1
OPERATOR: CLICKON cmd2	0.2
RETURN with goal accomplished	0.1
AG: select-fence	0.1
OPERATOR: MOVE to fence	1.1
OPERATOR: CLICKON fence	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
Unit task time	4.5
DECIDE: If more vents then GOTO STEP 1	0.1
RETURN with goal accomplished	0.1
Total time for three vents	42.39

AG: accomplish goal.

Table 8
The trace of the NGOMSL model of the preferred procedure for the panel clean-up task described in Bhavnani and John (2000)

TRACE of NGOMSL model for the preferred procedure of clean-up vent	Time (s)
AG: cleanup-vent	0.1
AG: cut-lines [[unit task]]	0.1
DECIDE: IF delete-part-of-element command not selected then: AG: setup-command-"delete-part-of-element"	0.1
AG: setup-command-"Delete-Part-of-Element"	0.1
OPERATOR: MOVE to <cmd1>	1.1
OPERATOR: CLICKON button	0.2
OPERATOR: MOVE to <cmd2>	1.1
OPERATOR: CLICKON button	0.2
OPERATOR: MOVE to <cmd3>	1.1
OPERATOR: CLICKON button	0.2
RETURN with goal accomplished	0.1
OPERATOR: MOVE to line-<n>	1.1
OPERATOR: CLICKON line-<n>	0.2
OPERATOR: MOVE to clip-point-<n>	1.1
OPERATOR: CLICKON line-<n>	0.2
RETURN with goal accomplished	0.1
Unit task time for first line	7.0
DECIDE: IF more lines to be cut, AG: cut-lines (for the next 10 lines the time cost of the DECIDE–AG: cut-lines cycles would be 3 s per line)	30
AG: extend-all-lines [[unit task]]	0.1
DECIDE: IF "Extend-to-Intersection" command not selected then: AG: setup-command-"Extend-to-Intersection"	0.1
AG: setup-command-"Extend-to-Intersection"	0.1
OPERATOR: MOVE to modify-cmd	1.1
OPERATOR: CLICKON modify-cmd	0.2
OPERATOR: MOVE to extend-cmd	1.1
OPERATOR: CLICKON extend-cmd	0.2
OPERATOR: MOVE to 1X-cmd	1.1
OPERATOR: CLICKON 1X-cmd	0.2
RETURN with goal accomplished	0.1
AG: extend-both-ends-of-each-cut-line	0.1
OPERATOR: MOVE to line-<o> (s1)	1.1
OPERATOR: CLICKON line-<o> (s2)	0.2
OPERATOR: MOVE to extend point for line-<o>	1.1
OPERATOR: CLICKON extend point	0.2
OPERATOR: MOVE to reset point for line-<o>	1.1
OPERATOR: CLICKON reset point	0.2
RETURN with goal accomplished	0.1
RETURN with goal accomplished	0.1
DECIDE if more lines then GOTO AG: extend-both-ends-of-each-cut-line else continue	0.1
Unit task time for 22 extensions (2 ends each for 11 lines)	103.1
RETURN with goal accomplished	0.1
Total time for 11 lines	140.3

AG: accomplish goal.

In answer to our second question, the analysis in Table 7 strikes us as more specialized than the one in Table 8. Beginning in the middle and shortening both ends of a line (the Table 8 procedure) strikes us as a procedure that could be used in manual drafting and many places in the CAD system. In contrast, "fencing" an object by selecting its vertices strikes us as much more specialized and, certainly, not something that would arise during manual drafting. Similar to the previous case of drawing complex shapes, deciding on whether to use the specific aggregate tool or cutting the lines one after the other depends on how many lines there are inside the panel. We argue that the local data of the specialized procedure were not strong enough to completely dominate over the use of the general procedure.

In answer to our third question, for the general procedure, the cut-line unit task entails moving to and clicking on each line twice, once to select it and once to indicate where it should be cut. In contrast, for the specialized procedure the aggregate unit task requires the architect to locate and click on vertices, decide whether the precise point was clicked, and verify that the "fence" snapped correctly into place. The first unit task of the specialized procedure takes 9.3 s to execute without any change to the external display state (only the internal system state is changed). On the other hand, the estimate of actual effort of 7 s for the cut-lines unit task is for the first line that is cut. As shown in Table 8, after the setup-command has been executed once, each subsequent cut-lines unit task requires only 2.9 s (plus 0.1 s for the DECIDE step). We therefore conclude that the general procedure is composed of more interactive components than the specialized procedure.

4.4. Summary of the reanalyses of the studies by Bhavnani and John: answers to the three questions

How does our reanalysis of Bhavnani and John's architects address our three questions? First, as Bhavnani and John showed, the recommended procedures are clearly more efficient than the procedures used. Second, our reanalysis suggests that the recommended procedures tend to be more specialized within the CAD system. In contrast, some of the preferred procedures are general procedures imported from the manual drafting task environment, whereas others seem composed of general unit tasks that can be used for many subtasks within the CAD system. Third, the general procedures started with unit tasks that were composed of more interactive actions. Similar to the analyses in Study I, the bias in the selection of these interactive unit tasks may lead to the strengthening of the general procedures. On the other hand, the local data of the specialized procedures were not strengthened enough to completely supplant the general procedures, even after years of experience. In contrast to Study I in which students had limited experience with HyperCard, in Study II the architects had years of experience with the CAD system. The architects clearly had sufficient knowledge of the recommended procedures. The paradox was that the inefficient procedures were still commonly used.

5. Study III: performance in a simulated VCR interface

Our explanations of the persistent inefficient behavior in the two data sets presented above partially rest on the assumption that people are biased to use general, interactive procedures. To provide further evidence to support our claims, we conducted an experiment in which we

created a general, interactive procedure that could be used for many purposes in the course of programming a simulated VCR. We also created a specialized procedure that would be the most efficient for a certain category of subtasks. We hypothesize that participants will be biased to use the general, interactive procedure even when it is less efficient than the specialized procedure.

5.1. Method

5.1.1. Participants and training

Sixteen students from Rensselaer Polytechnic Institute participated in the experiment. Before the experiment began, each subject was given a demonstration on how to use each of the procedures available to change the settings of the simulated VCR; this included an explicit demonstration of a specialized procedure (the Jump procedure discussed below). Subjects were then given a practice trial during which the experimenter answered any questions that they had. After subjects had successfully programmed the show in the practice trial, they were given eight more shows to program. The order of the shows was counterbalanced. All mouse-clicks were time-stamped to the nearest 16 ms and were saved to log files.

5.1.2. Procedure

Subjects were asked to program a simulated VCR (shown in Fig. 5) to record eight shows, each with a specific start time, end time, channel, and day-of-week. To begin a trial, subjects clicked on the "start" button on the screen. After this, the information needed to program the show could be accessed by clicking on the gray boxes in the show-information window below the simulated VCR. If needed, the information could be accessed throughout the trial by moving to and clicking on the gray boxes. To program a show, subjects clicked on the "program show" button in the simulated VCR, then clicked on one of the radio buttons to set the corresponding setting (e.g., each of the following eight fields; "start hour," "start 10 min," "start min," "end hour," "end 10 min," "end min," "channel," or "day-of-week"). After the radio button was clicked, the corresponding field in the display area would be highlighted.

After selecting the field to program, subjects could use a general procedure, clicking the *Up-Arrow* or *Down-Arrow* button to set the field. The initial values for all settings were "0." To use either the *Up* or *Dn* procedure, subjects simply moved the cursor to the Up-Arrow or Down-Arrow buttons, respectively, and continued clicking until the field setting matched the target setting. After all eight fields were set, subjects clicked on the "Record Show" button, then clicked on the "Stop" button to finish the trial. A feedback window provided information on trial duration and correctness. If all settings were correct, the subject would proceed to the next trial; if there was an error, subjects were required to program the show again.

The value of the channel setting could range from 0 to 63. For all shows the trial began with the channel set to 0. To set the channel, a specialized procedure was created that would jump the value of the channel field to 32. This *Jump* procedure required subjects to click on the "special" button on the simulated VCR (see Fig. 5), which popped up a dialog box to the right of the VCR. Subjects could then click on the "change setting to 32" radio button and the "OK" button to close the dialog box. After that, the channel setting on the VCR would be set to 32 and subjects could use the Arrow buttons to complete the setting.

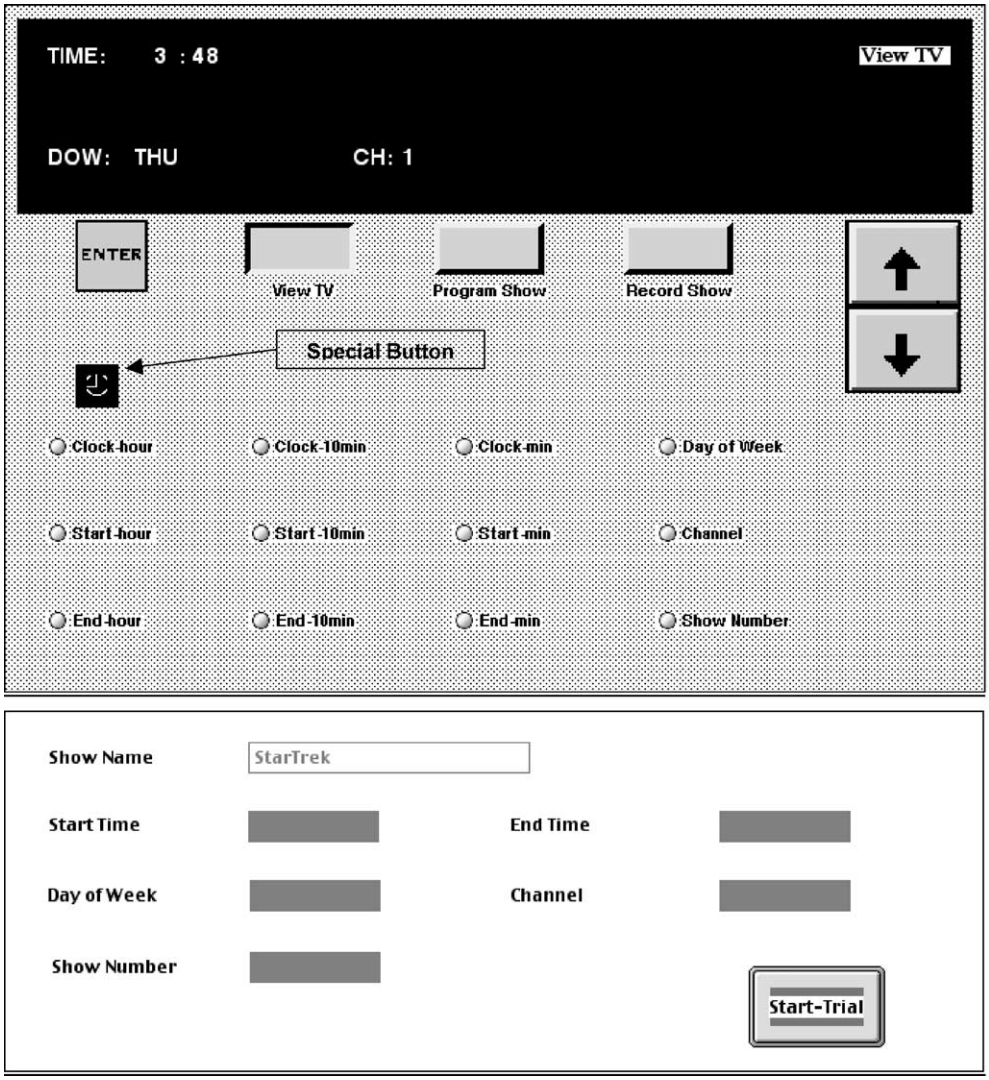

Fig. 5. The screenshot of the simulated VCR used in the experiment. Note the "special" button on the left of the display above the Clock-hour radio button and below the ENTER button.

Subjects programmed eight shows plus a practice show. For the practice show the value of the channel setting was 32. For the other eight shows the channel setting was 4, 16, 24, 28, 36, 40, 48, 60 (randomly sampled without replacement).

5.1.3. Comparing Up/Dn and Jump procedures

During the study, each subject received only one trial on each channel setting. Subjects were free to use Up, Dn, or Jump for any channel setting. However, it is clear that at best[8], each subject could only use one procedure for any given channel setting. Hence, it is difficult to empirically compare the same subject's performance across the three procedures for each channel setting.

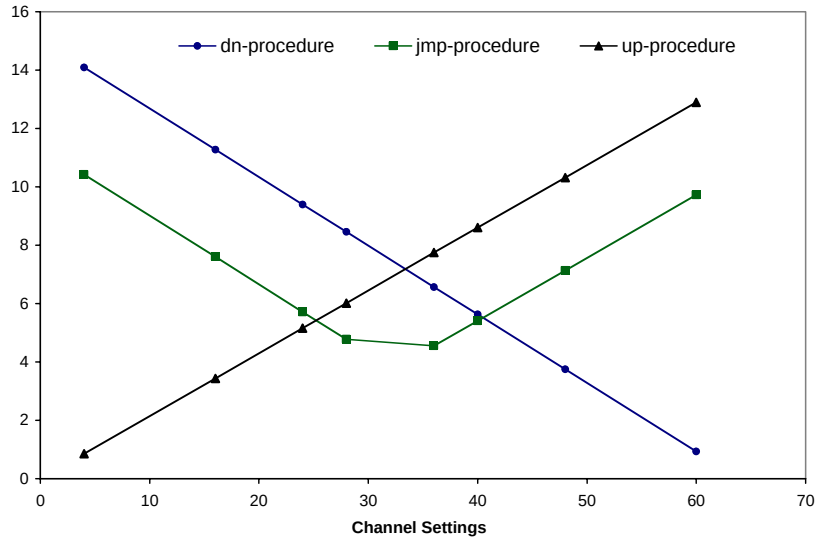

Fig. 6. The predicted times for the three procedures for setting the channel in different target channel values.

The design of the Jump procedure was guided by an NGOMSL analysis that allowed us to craft a procedure that would be faster for some channels, but not all. To show that the different procedures are differentially effective for different channels, we used data collected from each individual subject to derive a performance prediction for that subject for each procedure (Up, Dn, and Jump) at each channel setting. Like our NGOMSL analyses, these "Step" analyses provide an estimate of error-free, expert performance. However, the inter-keystroke times for each step type for each subject is based on the mean time, for that subject, across all steps of that type throughout all eight shows.

We performed a Step Analysis for the Up, Dn, and Jump procedures. In each case, the Step Analysis begins with the first button clicked after clicking on the channel radio button, and ends at the button click before clicking on the Enter button. Four inter-keystroke times were estimated from the empirical data—Up-Up (Up-arrow to Up-arrow button), Dn-Dn (Down-arrow to Down-arrow button), Special-Up (Special, Channel-32, OK, to Up-arrow button), and Special-Dn (Special, Channel-32, OK, to Down-arrow button). By plugging in the empirical estimates, this analysis allows us to predict the time required for each subject to set the channel to different values using each of the procedures. The mean of the estimates for all 16 subjects is shown in Fig. 6. The Jump procedure is the most efficient in channel settings "28," "36," and "40"; the Up procedure is the fastest in "4," "16," and "24"; and the Dn procedure is the fastest in "48" and "60."

Since only the Up/Dn procedures are used to program all other settings except the channel setting, the Up/Dn procedures are clearly used more frequently and are more general than the Jump procedure. Also, since each button press in the Up/Dn procedure increments or decrements the setting as reflected on the external display, the Up/Dn procedures are also clearly more interactive than the Jump procedure.

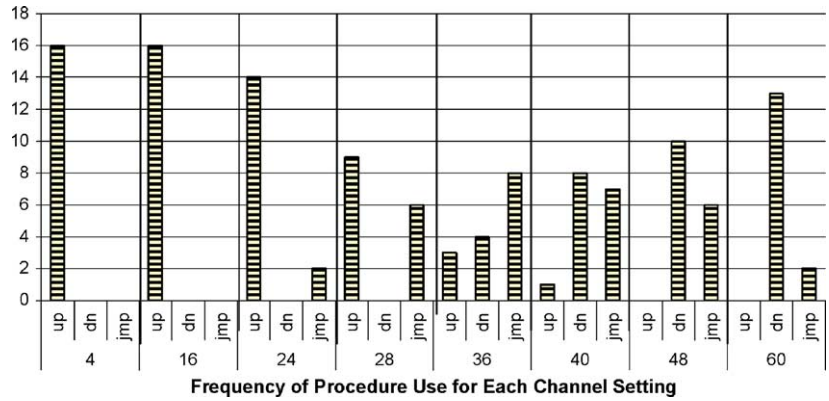

Fig. 7. The frequencies of use for each of the procedures for setting the channel in the simulated VCR (max = 16 per cell–3 data points missing—see text).

5.2. Results

5.2.1. Frequency of use of the specialized and general procedures

Each of the 16 subjects correctly programmed eight trials for a total of 128 trials. For three of these trials the log files did not clearly indicate whether the procedure used by the subject was predominately Up or Dn (however, it clearly was not Jump). Our interpretation of the log file is that the subjects may have initially been trying to program the wrong channel setting. In any case, we excluded these trials from the analyses reported below. Fig. 7 shows the frequency of use of each procedure for each channel setting. An analysis of variance (ANOVA) yielded a significant main effect of procedure, $F(2, 30) = 3.64, p < .04$, with the Up being used more frequently (0.46) than either the Dn (0.27) or Jump (0.24) procedure. As expected, the interaction of Channel Setting by Procedure was also significant, $F(14, 210) = 23.24, p < .0001$.

For our purposes, the key channel settings are channels 28, 36, and 40. For these settings the Step Analysis showed that Jump would be more efficient than either Up or Dn. As shown by Fig. 7, for channel 28 the specialized Jump procedure was used by 6 of 15 subjects (with one data point missing). For channel 36, Jump was used by 8 of 15 subjects. Finally, for channel 40, Jump was used by 7 of 16 subjects.

The experimental data replicate under laboratory conditions *the paradox of the active user*. Subjects often fail to choose the most efficient method. Rather, in cases where the specialized method is more efficient than the general method, the specialized method is used less than half of the time. Even for the channel values of “28” and “36,” only about half of the subjects chose to use the special procedure. The patterns of results support our hypothesis that subjects were biased to use the more general, interactive procedures.

5.3. Summary and discussions of the data from the simulated VCR study

Consistent with the analyses of the two data sets, we found that subjects were biased to use general, interactive procedures, even when they were inefficient. The Up/Dn procedures

were frequently used throughout the simulated VCR; hence, it is clearly more general than the Jump procedure. The general procedure was also more interactive than the specific procedure. Given the design of the experiment, the more specific Jump procedure was used less frequently than the general procedures. This feature of the design, as well as the selection bias to the general, interactive procedure, lead to a slow strengthening of the local data of the specialized procedure. As a result, the specialized procedure, even when it is more efficient, never completely dominated over the general procedures even after extensive experience.

6. Conclusions and discussions

For each of the three data sets we asked and answered three questions in the affirmative. First, we found that sometimes people used preferred procedures that were less efficient than the recommended procedures; that is, we documented *the paradox of the active user* (Carroll & Rosson, 1987). Second, it was the case that the recommended procedures were more specialized than the preferred procedures, and in many cases, the recommended procedures were the most efficient only under specific conditions. Third, compared to the recommended procedures, at least the initial stages of the preferred procedures tended to be composed of interactive actions that result in rapid, incremental feedback from the external display. These regularities lead us to conclude that the paradox of the active user cannot be simply cast as *performance error*, but as a result of the underlying cognitive process of selecting procedures.

In all three data sets we did not attempt to analyze the effect of generality and interactivity orthogonally, thus their effects on the selection of procedures are often confounded. Although we do not have enough data to tease apart their effects, at least we can conclude that, with all else equal, a procedure that has either one of the properties will be used more frequently than a procedure that is specialized and non-interactive. Moreover, from our analyses, it seems plausible that these two properties frequently co-occur in real-world situations. Instead of analyzing data from natural settings, one of our future plans is to further our understandings by studying the individual effects of generality and interactivity on selection of procedures in a series of well-controlled experiments.

The preferred procedures from our first two data sets were only a small percentage of all procedures used by the users. We showed that the preferred procedures were inefficient, more general and interactive than the corresponding recommended procedures. This allowed us to provide an explanation for the paradox of the active user. However, we do not mean to imply that all general procedures are inefficient or that specialized procedures are always not preferred. In fact, it is quite plausible that for other tasks in different task environments, at least some of the time, general procedures would be the most efficient, or that specialized procedures would be preferred to more general "recommended procedures." However, for the first case, the most efficient general procedures would not present a paradox and, hence, would not be a focus of the current paper. For the second case, our analyses would have caught them if they existed, but at least in our data sets we did not find any preferred procedures that were more specialized than the recommended procedures. We conclude that, given the three diverse sets of results, our analyses point toward general characteristics of preferred procedures.

Carroll and Rosson observed that the paradox of the active user is composed of a production bias and an assimilation bias. Production bias refers to the observation that users are often not discovering and using functions that could have been more efficient. Assimilation bias refers to the observation that users often figure out how to use what they already know to achieve new goals. Their observations are in general consistent with our analyses, except that inefficient procedures seem to persist even when users apparently have knowledge of more efficient procedures. Our results allow us to define these biases in terms of cognitive processes—assimilation bias can be explained in terms of the strength of global data in general procedures; production bias can be explained by the advantage of interactive actions that offload cognitive effort to external display for better performance monitoring and error control. Although there may be other reasons for the persistence of inefficient behavior, our contribution lies in providing an explanation of the paradox at the level of cognitive processes.

6.1. Normative rationality and stable suboptimality

The idea of optimization, deeply rooted in common sense, holds behavior to be normatively rational. In fact, we believe that the reason why Carroll and Rosson used the phrase "paradox of the active user" to describe the persistent use of inefficient procedures was perhaps based on the common belief that with enough practice, behavior will eventually stabilize on the optimal level of performance. Although the principle of optimality has been useful as an explanation of behavior, we share our view with others (e.g., see Chapter 10 of Herrnstein, 1991) that a more general class of explanations is the principle of stability. In fact, the principle of stability is a necessary but not a sufficient condition for the principle of optimality, as systems that follow the principle of optimality also follow the principle of stability, but a stable level of performance is not necessarily the globally optimal. In analyzing our data sets, our assumption is that the local adaptive process of procedure selection has led to a stable, but suboptimal, level of performance.

From our analyses, people chose to use suboptimal procedures even when they apparently had knowledge of the optimal procedures, and thus have violated the normative principle of rationality. Our explanation is that the lack of strong local data for specialized procedures in interactive environments implies that considerable cognitive effort is required to mentally look-ahead and process task-specific information to determine which procedure is the optimal procedure. Our analyses also show that the optimal procedure often starts with "set-up" unit tasks that change only the internal states of the artifact, not the external display. On the other hand, general procedures that start with interactive actions require less cognitive effort as each action results in an immediate change to the external display that, in turn, cues the next action. We interpret the selection bias as indicating that the selection process is myopic—that is, only the initial unit tasks are compared, not the whole procedure. The myopic nature of the selection process has been observed in animals (Vaughan, 1981, 1985) and humans (Herrnstein, 1991). If only the initial unit tasks are compared, the bias to use general interactive unit tasks may be explained by the low cognitive effort involved when incremental feedback provided by the external display is utilized as cues for actions. Unfortunately for the users, we observed that the bias to use the interactive unit task leads to a path that requires more effort in the long run. However, it seems that each of the myopic choices may still have a flavor of rationality

(Anderson, 1990; Simon, 1956) to it so that, from a global perspective, the myopic nature of the selection process has led to a stable, rather than a globally optimal, cost–benefit tradeoff between procedures.

Our explanation assumes that people seldom make a once-and-for-all decision on procedures. In fact, interactive behavior seems to be composed of a series of distributed choices. We propose that selection of inefficient procedures is a result of a myriad of myopic, often innocent, choices. Indeed, anecdotal evidence suggests that users often notice the existence of a more efficient procedure only when the task has already been accomplished by a less efficient procedure. Our analyses suggest that a series of biased selection of interactive unit tasks often lead to a stable suboptimal level of performance, as measured by a global, normative standard of rationality.

Notes

1. In the original ASCM model, there is an intermediate layer called the featural data defined between the layers of global and local data. Since we do not have the same level of analysis in this paper, we merge the layers of featural and local data in the original ASCM and refer to it as local data throughout this paper.
2. In this example, since the center-text procedure is highly specialized, its local and global data are identical. However, this is not necessarily true for less specialized procedures.
3. Given two unit tasks with exactly the same effectiveness and efficiency, a biased selection is defined as the selection of one of the unit tasks more than 50% of the time.
4. HyperCard uses the metaphor of creating a stack of cards. Individual screens are referred to as “cards.” The screens that constitute the program are referred to as a “stack.”
5. An appendix containing analyses of other recommended and preferred procedures may be downloaded from the annex maintained by the Cognitive Science Society, see http://cogsci.psy.utexas.edu/supplements/. The same material is also available from the Publisher’s website.
6. The lasso tool is intended to remind users of the rope used by cowboys to “lasso” or rope cattle. Using the tool requires dragging the mouse around the to-be-selected object.
7. Many thanks to Suresh Bhavnani for answering our many questions and supplying us with his NGOMSL analyses, which we modified to produce our analyses in Tables 7 and 8. The modifications impose the conventions of our NGOMSL analyses onto the analyses supplied by Bhavnani.
8. At “best”–at worst the subject mixed the channel setting procedure with random keys that prevented us from cleanly assigning performance time to a given procedure.

Acknowledgments

The work reported was supported by grants from the Air Force Office of Scientific Research AFOSR#F49620-97-1-0353 and F49620-03-1-0143. We thank Chris Meyers for conducting

Study III. We also appreciate the effort and many useful suggestions by John Anderson, Peter Cheng, and two anonymous reviewers.

Appendix A. Preferred procedures in the first data sets

Qualitative descriptions of the preferred procedures are presented in this appendix. For each preferred procedure, the analysis is consistent with the three major claims of the paper: (1) the recommended procedure was more efficient than the preferred procedure, (2) the recommended procedure involved more specialized unit tasks than the corresponding preferred procedure, and (3) the initial unit tasks in the preferred procedure were composed of more interactive actions. NGOMSL models of the procedures are available from the authors upon request.

A.1. Create new button just to change mode

HyperCard supports several procedures of changing to button-mode. Clicking on the button icon in the tool palette is the textbook way. An alternative is to use the menu to select the item "new button." This selection puts a new button on the card and changes the mode to button-mode. Note that if the card needs a button, then using the menu to create a new button and to change to button-mode is the recommended procedure. However, two students in BM1 used this procedure to change to button-mode even though all buttons needed for a card had already been created. Hence, at some later point, the extra, *change mode*, buttons were deleted. The preferred procedure is clearly less efficient than the recommended procedures, since one has to delete the extra button later.

The rival procedures both strike us as specialized to the HyperCard environment. According to the NGOMSL analysis, the actual effort for the create-a-new-button unit task is slightly higher than that for the select-<button-mode> unit task. However, in the cases we observed where this preferred procedure was used, the student had recently created a button. Hence, the create-a-new-button unit task may be more active. By BM2, no student used this unit task to change mode.

A.2. "Deleted" button by covering it with a white patch

One student "deleted" an extra button by covering it with a white rectangle. This preferred procedure seemed to be transferred from the natural environment—paint over something that is no longer needed. Although this preferred procedure was almost as efficient as the recommended procedure (which requires pressing the delete key on the keyboard), it created potential extra effort later in the task because the white patch sometimes went one layer below the button and the button was shown again. When the button was shown, the student had to click on the white patch to bring it one layer above the button to cover the button again. Since the button is a specific object in HyperCard, the recommended procedure was obviously more specific than the preferred procedure, which is a simple and general procedure in most natural envi-

ronment. In this case, perhaps both procedures are equally interactive as only a single action is required.

A.3. Use spacebar to center text in a field box (BM2) or button name on a button (BM1)

Through BM1, the only text tool available to the students was the bitmapped text tool. To move text around, students could type and then select and drag the text (as discussed in the body of the paper) or they could insert leading blank spaces to move the start of the text to the right. During BM1 one student attempted to insert leading spaces into the button label box in the button tool palette. This preferred procedure is clearly negative transfer from word processors or even traditional type-writer. HyperCard does not allow blank spaces in the button label box so this procedure was unsuccessful. During BM2 another student attempted to use blank spaces to center text in a HyperCard Field Box. Field boxes support ASCII (not bitmapped) text and come with the regular range of alignment adjustments, including centered.

For BM1 there was no recommended procedure for accomplishing what the student was trying to do. Hence, it was reasonable for the student to attempt to invent a procedure. By the time of BM2, the students had learned more efficient, but specialized procedures for manipulating text in field boxes. However, we would have expected the procedure to be more general and used more frequently than the more specialized recommended procedure.

A.4. Create buttons by copy-paste

In button mode it was possible to select a button by clicking on it with the mouse, copy it, and then paste it at another spot on the current card or on another card. This copy-paste unit task seemed to be simple and common across many computer interfaces. Indeed, for the Macintosh, the copy and paste commands used the same keystrokes to copy and paste a button as to copy and paste text and graphics. Unfortunately, as the links associated with the buttons were copied too, the students had to change the links and labels of each of them, making this preferred procedure less efficient than the recommended procedure of creating a new button. We argue that the student had used the general copy-paste procedure acquired outside of HyperCard instead of the specific copy-paste procedure in HyperCard or the more specialized menu objects required to create a new button.

A.5. Attempt to type bitmapped text into ASCII field box and attempt to type ASCII text into bitmapped rectangle (two procedures)

Bitmapped rectangles require a different set of actions than do field boxes. Bitmapped text inside bitmapped rectangles, once created, cannot be changed. In contrast, text inside field boxes is editable. Bitmapped text is created in text mode, but field box text is created in browse mode. However, the displays of bitmapped rectangles and field boxes are identical (with the exception that in field box mode there are lines inside the field boxes).

Indeed, we found that there was much confusion of these two ways of displaying text. After creating the field box, three students selected text mode and typed bitmapped text inside the field box. As with the attempt to type the label on the button, the result was that the text was

typed on the card, not in the object—in this case, not in the field box. Recovering from this mistake required students to erase the text they typed, change to browse mode, and type again. One student selected browse mode and attempted to type text into a bitmapped rectangle, but this attempt failed. Since field boxes were used only in BM2, no confusion of this type was found in BM1. This confusion shows that the students failed to apply the right procedure to the right objects on the screen, although the common unit task of type-text was correctly selected in both cases. The distinction between bitmapped text and ASCII text is specific to HyperCard. Students were confused when they could not select the right specific unit tasks.

A.6. Copy-paste-edit text instead of creating new editable text

In BM2, rather than typing new restaurant information directly into each field of a new restaurant card, S4 first copied and then pasted text from another card. This student then clicked on and edited the text. This preferred procedure had roughly the same efficiency as the recommended procedure (the exact comparison cannot be easily made as it depends on the content of the text). This case is therefore perhaps not a "real" paradox. On the other hand, the copy-paste procedure is applicable to all objects, however, not all text are editable in HyperCard. It is therefore plausible that the copy-paste procedure is more general than the create-editable text procedure. We suspect that this student wanted to make sure that editable text was copied, and did not want to directly type text inside the box (the typing procedure is the same for bitmapped text and editable text).

A.7. Draw button

Not all preferred procedures were successful. The recommended procedure to create a button is to go to the "Objects" menu and select "New Button." However, S4 and S8 tried to use the draw tools to draw the buttons on the cards. It is possible that students fail to remember the recommended procedure, and the drawing of the buttons allowed the students to directly create the object of interest. Obviously, this is a procedure that is commonly used in the natural environment. However, the interface did not allow buttons to be created this way, and the students ended up erasing the "button" and using the menu to create a "real" button. This example demonstrates that during this early stage of learning, the use of general procedures acquired from another environment could eventually lead to inefficient procedures at later stage of learning. This procedure also seems more interactive than the recommended procedure that requires the use of the pull-down menu.

A.8. Use lasso to select button or to select field

In BM1, S5 attempted to select a button by using the lasso tool. In BM2, S6 attempted to use the lasso tool to select a field. In both cases, the recommended procedure required the student to first select the proper mode (button mode or field mode) from the tools palette and then to select the object of interest by clicking on it. In both cases, when the object was finally selected using the recommended procedure, it became clear that the student's goal was to reposition the object. The lasso tool is a general procedure that applies to almost any objects except buttons

and fields. In this case, both students would have used this procedure to move graphics and bitmapped text. Hence, when the recommended procedures and the preferred procedures were competing, S5 and S6 selected the more general lasso tool preferred procedure.

A.9. Use eraser to delete button

One student selected the eraser from the draw-tools palette and attempted to use it to delete a button. This would-be preferred procedure works well with graphics and bitmapped text, but not with buttons or fields. This student eventually "deleted" the button by covering it with a white patch (see its description above). For these students, deleting a button or a field was a much rarer occurrence than deleting a graphic or bitmapped text. Hence, it is reasonable to assume that the general procedure of using the eraser was used more often in different contexts (even in the natural environment) than the more specialized procedure that was required to remove a button.

A.10. "Deleted" button by covering it with a black patch

The student, who successfully eliminated (from view) a button by covering it with a white patch, initially tried to cover it with a black patch. Although the black patch did cover the button, it left a black square on the card. However, this seems to have been a necessary step in the invention of the successful procedure. The student was clearly trying to find something from the draw palette that would obscure the button. S1 realized that she or he could draw squares that were filled with a pattern or a color and that these squares could be placed on top of the button. (Note that this prolonged problem solving involved in an attempt to invent a procedure to get rid of a button, supports our contention that the recommended procedure was specialized and not well-practiced.)

References

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R. (2002). Spanning seven orders of magnitude: A challenge for cognitive modeling. *Cognitive Science*, *26*(1), 85–112.

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, *42*, 7–49.

Anderson, J. R. & Lebiere, C. (Eds.). (1998). *Atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Beekman, G. (1991). *HyperCard 2 in a hurry*. Belmont, CA: Wadsworth Publishing Company.

Bhavnani, S. K. (2000). *Designs conducive to the use of efficient strategies, designing interactive systems: Processes, practices, methods, and techniques* (pp. 338–345).

Bhavnani, S. K., & John, B. E. (1996). Exploring the unrealized potential of computer-aided drafting. In M. J. Tauber, V. Bellotti, R. Jeffries, J. D. Mackinlay, & J. Nielsen (Eds.), *ACM CHI'96 Conference on Human Factors in Computing Systems* (pp. 332–339). New York: ACM Press.

Bhavnani, S. K., & John, B. E. (1997). From sufficient to efficient usage: An analysis of strategic knowledge. In S. Pemberton (Ed.), *ACM CHI'97 Conference on Human Factors in Computing Systems* (pp. 91–98). New York: ACM Press.

Bhavnani, S. K., & John, B. E. (1998). Delegation and circumvention: Two faces of efficiency. In C.-M. Karat, A. Lund, J. Coutaz, & J. Karat (Eds.), *ACM CHI'98 Conference on Human Factors in Computing Systems* (pp. 273–280). New York: ACM Press.

Bhavnani, S. K., & John, B. E. (2000). The strategic use of complex computer systems. *Human–Computer Interaction*, *15*(2/3), 107–137.

Bhavnani, S. K., John, B. E., & Flemming, U. (1999). The strategic use of CAD: An empirically inspired, theory-based course. In M. G. Williams, M. W. Altom, K. Ehrlich, & W. Newman (Eds.), *ACM CHI'99 Conference on Human Factors in Computing Systems* (pp. 183–190). New York: ACM Press.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Carroll, J. M., & Rosson, M. B. (1987). Paradox of the active user. In J. M. Carroll (Ed.), *Interfacing thought: Cognitive aspects of human–computer interaction*. Cambridge, MA: MIT Press.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, *5*, 121–152.

Doane, S. M., Pellegrino, J. W., & Klatsky, R. L. (1990). Expertise in a computer operating system: Conceptualization and performance. *Human–Computer Interaction*, *5*(2/3), 267–304.

Ericsson, K.A. (1974). *Problem-solving behaviour with the 8-puzzle II: Distribution of latencies. Report No. 432*. Department of Psychology, University of Stockholm.

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science*, *24*(2), 205–248.

Gray, W. D., & Anderson, J. R. (1987). Change-episodes in coding: When and how do programmers change their code? In G. M. Olson, S. Sheppard, & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop* (pp. 185–197). Norwood, NJ: Ablex.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world performance. *Human–Computer Interaction*, *8*(3), 237–309.

Herrnstein, R. J. (1991). Experiments on stable suboptimality in individual behavior. *The American Economic Review*, *81*(2), 360–364.

Kieras, D. E. (1997). A guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), *Handbook of human–computer interaction* (2nd ed., pp. 733–766). New York: Elsevier.

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science*, *18*(4), 513–549.

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from tower of Hanoi. *Cognitive Psychology*, *17*, 248–294.

Larkin, J. H. (1989). Display-based problem solving. In D. Klahr & K. Kotovsky (Eds.), *Complex information processing: The impact of Herbert A. Simon* (pp. 319–341). Hillsdale, NJ: Lawrence Erlbaum Associates.

Mayes, J. T., Draper, S. W., McGregor, A. M., & Oatley, K. (1988). Information flow in a user interface: The effect of experience and context on the recall of MacWrite screens. In D. M. Jones & R. Winder (Eds.), *People and computers IV* (pp. 275–289). New York: Cambridge University Press.

Nilsen, E. L., Jong, H., Olson, J. S., Biolsi, K., Rueter, H., & Mutter, S. (1993). The growth of software skill: A longitudinal look at learning & performance. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), *ACM INTERCHI'93 Conference on Human Factors in Computing Systems* (pp. 149–156). New York: ACM Press.

O'Hara, K. P., & Payne, S. J. (1998). The effects of operator implementation cost on planfulness of problem solving and learning. *Cognitive Psychology*, *35*, 34–70.

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

Payne, J.W., Bettman, J.R., & Johnson, E.J. (1993). *The adaptive decision maker*. New York: Cambridge University Press.

Payne, S. J., Richardson, J., & Howes, A. (2000). Strategic use of familiarity in display-based problem solving. *Journal of Experimental Psychology-Learning Memory and Cognition*, *26*(6), 1685–1701.

Siegler, R. S., & Lemaire, P. (1997). Older and younger adults' strategy choices in multiplication: Testing predictions of ASCM using the choice/no-choice method. *Journal of Experimental Psychology—General*, *126*(1), 71–92.

Siegler, R. S., & Shipley, C. (1995). Variation, selection, and cognitive change. In T. Simon & G. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling* (pp. 31–76). Hillsdale, NJ: Lawrence Erlbaum Associates.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, *63*, 129–138.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "broke"? A model of learning the past without feedback. *Cognition*, *86*(2), 123–155.

Vaughan, W., Jr. (1981). Melioration, matching, and maximization. *Journal of the Experimental Analysis of Behavior*, *36*, 141–149.

Vaughan, W., Jr. (1985). Choice: A local analysis. *Journal of the Experimental Analysis of Behavior*, *43*, 383–405.