

The Nature and Processing of Errors in Interactive Behavior

WAYNE D. GRAY

George Mason University

Understanding the nature of errors in a simple, rule-based task—programming a VCR—required analyzing the interactions among human cognition, the artifact, and the task. This analysis was guided by least-effort principles and yielded a control structure that combined a rule hierarchy task-to-device with display-based difference-reduction. A model based on this analysis was used to trace action protocols collected from participants as they programmed a simulated VCR. Trials that ended without success (the show was not correctly programmed) were interrogated to yield insights regarding problems in acquiring the control structure. For successful trials (the show was correctly programmed), steps that the model would make were categorized as matches to the model; steps that the model would not make were violations of the model. The model was able to trace the vast majority of correct keystrokes and yielded a business-as-usual account of the detection and correction of errors. Violations of the model fell into one of two fundamental categories. The model provided insights into certain subcategories of errors; whereas, regularities within other subcategories of error suggested limitations to the model. Although errors were rare when compared to the total number of correct actions, they were important. Errors were made on 4% of the keypresses that, if not detected, would have prevented two-thirds of the shows from being successfully recorded. A misprogrammed show is a minor annoyance to the user. However, devices with the approximate complexity of a VCR are ubiquitous and have found their way into emergency rooms, airplane cockpits, power plants, and so on. Errors of ignorance may be reduced by training; however, errors in the routine performance of skilled users can only be reduced by design.

I. INTRODUCTION

Overview

The exact sequence of behavior that is required to program a VCR and other simple rule-based devices might seem arbitrary as long as it does not result in overt errors.

Direct all correspondence to: Wayne D. Gray, Human Factors & Applied Cognitive Program, Department of Psychology, George Mason University, Fairfax, VA 22030; E-Mail: gray@gmu.edu

However, although many patterns of interaction often are possible, few patterns are typically employed. Performance is constrained by the interactions between the task the user is attempting to accomplish, the design of the artifact (i.e., tool or device) used to accomplish the task, and human cognitive, perceptual, and motor operations. What emerges from this interaction of embodied cognition with task and artifact is the *interactive behavior* needed to perform a given task on a given device.

A cognitive model can be built that will generate the correct interactive behavior for using the artifact to accomplish the task. Human behavior can be characterized as *matching* or *violating* the model's behavior. Most obviously, the model should be capable of tracing any sequence of correct behavior, but should fail to trace erroneous behavior. However, beyond this, the model is the key to understanding the nature, detection, and correction of errors in rule-based tasks. Without a detailed model, the classification of errors must be limited to simple categories such as mistakes, false alarms, or slips. With a detailed model, the vocabulary available to the analyst expands and an explanation of the errors may be given in terms of the model's control structure.

Besides overt errors, the model may fail to trace other behaviors. For example, suppose the user's goal is to set a VCR to record StarTrek; a show that airs Saturday on channel 11. The user toggles the day-of-week from Tuesday (today) to Thursday, toggles the channel from 4 to 11, and returning to day-of-week, toggles it from Thursday to Saturday. Furthermore, suppose that in switching from day-of-week to channel to day-of-week no button presses were required other than what would have been used if the user completed the day-of-week programming (from Tuesday to Saturday) before setting the channel. Although this switching behavior is puzzling, as no overt error occurred, it is difficult to characterize exactly why it is puzzling. The model proposed in this paper can be used to solve this puzzle by localizing the nature of the violation and characterizing its cognitive cost.

Rule-Based Tasks

The interactive behavior for programming the VCR can be considered rule-based and, as such, is similar to that used in the performance of a large class of skilled activities. Rule-based behavior is typically contrasted with problem solving; the distinction is best viewed as a continuum whose dimension is the amount of *search control knowledge* used to guide the search through a problem space (Card, Moran, & Newell, 1983; Rasmussen, 1983). Rule-based performance may emerge in two ways. The first involves extended experience and is the domain of studies of expertise. Problems that are extremely complex for the novice often become simple, straight forward, and rule-based for the expert. Some facets of, for example, computer programming, medical diagnoses, algebra, and chess (Ericsson & Lehmann, 1996; Ericsson & Smith, 1991) fit this description. The second way that rule-based performance may emerge is by design; specifically, by designing artifacts to accomplish a circumscribed set of tasks by following a limited number of rules.

Following Fitts (1967), theories of skill acquisition typically distinguish among three stages (see, e.g., Anderson, 1995; VanLehn, 1996). By Anderson's description, stage one

involves acquiring the declarative knowledge needed to perform the task. In stage two this declarative knowledge is organized into the rules (procedures) used for the task. During stage three, applications of the rules become more rapid and require fewer cognitive resources.

Many devices that are used daily are programmed only occasionally—examples include; clock radios, preset stations in car radios, programmable thermostats, and VCRs. It seems fair to loosely characterize our knowledge of such devices as *somewhere around the beginning of stage three*. Hence, the massed practice with the VCR given to participants in the current study has arguably brought them to a knowledge state that is similar to that gained from more casual and more occasional encounters with rule-based interactive devices in daily life. In this study, a participant's success in programming a show was interpreted as evidence that the participant had acquired the rules needed to program the VCR. Having acquired the rules, the participants were somewhere around the beginning of stage three.

Errors in Rule-Based Tasks

Errors. Over the last 20 years, there have been flurries of interest in the issue of errors. Some researchers have attempted to build theoretical classifications that incorporate a wide range of errors from a wide range of tasks. Among the most influential of these approaches are Norman (1981) and Reason (1990). Most studies have been more limited, attempting to explain how a certain cognitive mechanism or process could lead to errors in a particular task (examples include Anderson & Jeffries, 1985; Booth, 1990; Brown & VanLehn, 1980; Brown & Gould, 1987; Cuniff, Taylor, & Black, 1989; Huguenard, Lerch, Junker, Patz, & Kass, 1997; Johnson et al., 1981; Lansdale, 1995; Lerch, Mantei, & Olson, 1989; Rizzo, Bagnara, & Visciola, 1988; Seifert & Hutchins, 1992; Smelcer, 1995; Xiao, Mackenzie, & Group, 1995; Young & Whittington, 1990).

The studies that are most congenial to the current approach are those that, in the confines of one paradigm, have carefully collected errorful and error-free behavior and then subjected it to a fine-grained analysis. These include Nooteboom's (1980) studies of errors in linguistic performance, VanLehn's studies of bugs in children's subtraction (VanLehn, 1990), and Allwood's studies of statistics and programming (Allwood, 1984; Allwood & Bjorhag, 1990; Allwood & Bjorhag, 1991). The results of these studies shed as much light on correct, error-free performance as on the nature of errors.

Rule-Based Tasks. The errors of interest in the current paper are errors in the execution of routine, rule-based tasks. In 1983, Card, Moran, and Newell proposed GOMS¹ as an approach to the analysis of skilled behavior. For GOMS, skilled behavior consisted of goals and operators for accomplishing those goals. For Card et al., a hallmark of skilled behavior was that sequences of subgoals and operators could be regarded as methods. The steps of a method are executed in sequential order and without deliberation. From the perspective provided by this framework, Card et al., stated that the "detection and correction of errors is mostly routine" (Card et al., 1983, p. 146). However, they

recognized that the rigid control structure of GOMS was inadequate for a general treatment of errors and that “a more general control structure [was] required” (p. 147).

Perhaps the research community was discouraged by this dismissal of the detection and correction of errors in rule-based behavior as “mostly routine.” Perhaps Card et al. raised the bar too high by pointing out that the study of errors in routine, rule-based tasks needed to be based on a better understanding of the control structure of such tasks. For whatever reason, little research has focused on errors in this important category of human behavior. For example, in a review written in 1990, Olson and Olson could report only two studies that examined errors in routine performance (these were Lerch et al., 1989; Smelcer, 1995). Seven years later, John and Kieras (1996) concluded that, “No methodology for predicting when and what errors users will make as a function of interface design has yet been developed and recognized as satisfactory . . . even the theoretical analysis of human error is still in its infancy.”

The most important exception to the general lack of progress in understanding errors in rule-based tasks is an exception that proves the rule. Grounding their work in a control structure adequate to perform a simple rule-based task, Byrne and Bovair (1997) recently introduced the concept of postcompletion errors. Examples of postcompletion errors include leaving the card in the ATM after getting money or leaving the original inside the copy machine after making copies. Postcompletion errors are the omission of device-specific goals (e.g., the need to get your card back from the ATM after getting money) after the completion of task-based goals (e.g., getting money).

Preview of the Rest of the Article

The current effort has three parts. First, a data set of performance was built by collecting action protocols² of participants as they programmed a simulated VCR. This data set was divided into two parts: trials that ended without the show being successfully programmed versus those in which it was. The main product of this part was a keypress³ taxonomy that classified all keypresses as hits or errors.

Second, the interactive behavior required to use the VCR to record a show was analyzed. The assumption guiding this analysis was that cognition is rational in that, with all else equal, it tends to maximize benefits while minimizing costs (see, e.g., Anderson, 1990; J. Payne, Bettman, & Johnson, 1993). Three principles of cognitive engineering were suggested by this analysis: least-effort in operating the device, least-effort in mapping prior knowledge to device knowledge, and least-effort in place-keeping.

The first two principles influence the sequence of steps taken to program the VCR. The third yields a display-based approach to difference-reduction (also called hill-climbing) that catches discrepancies between the VCR's settings and four task-based goals (the show's start time, end time, day-of-week, and channel) that are maintained in memory.

A model based on this analysis was built that supports simple programming for shows. Having built the model, an unexpected outcome was that display-based difference-reduction facilitated the detection and correction of errors. Indeed, for the particular VCR studied, detection and correction of errors reflected the business-as-usual operation of the

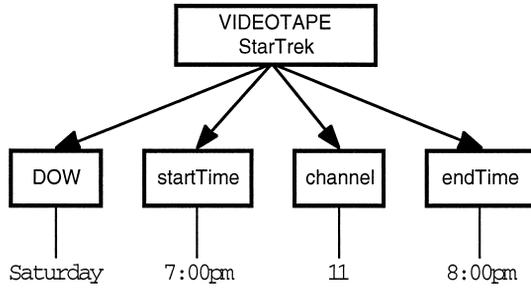


Figure 1. Task-based goal hierarchy for videotaping a show.

model. Discrepancies between the current state of the VCR and one of the four task goals were resolved in a like manner regardless of how the discrepancy arose.

Third, a goal push/pop taxonomy was created and the model was used to trace participant performance. Each push and pop that the model could trace was recorded in the taxonomy as a hit. Any push or pop that the model could not trace was recorded as a violation of the model's control structure. The categorized data were interrogated to yield information regarding the nature, detection, and correction of errors.

II. THE EMPIRICAL STUDY

The user's task is to record a show that is to be broadcast on a given day-of-week, from a given start time, to a given end time, on a given channel. This simple, task-based hierarchy is shown in Figure 1.⁴

In addition to knowing what needs to be done, the user must have knowledge as to how to do it using the artifact shown in Figure 2.

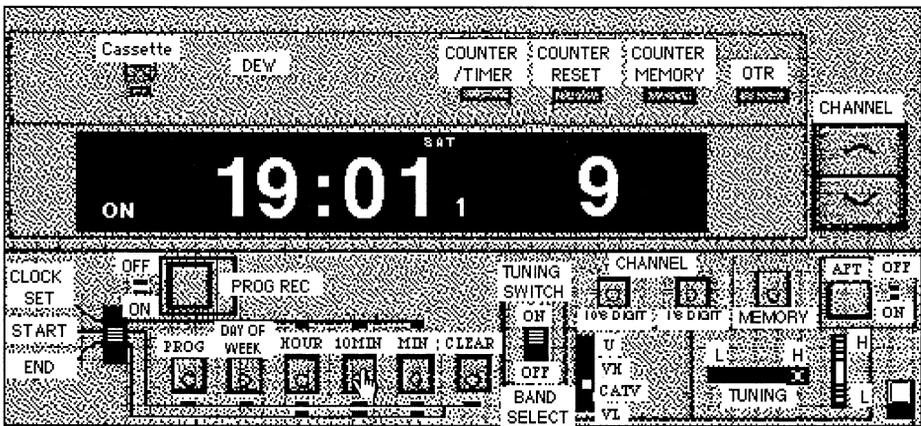


Figure 2. VCR during programming.

Methodology

Participants

Nine participants (six females and three males) were recruited from the students and staff of Fordham University. All participants were volunteers who participated for no pay or other compensation. The average age was 32 with a range from 23 to 45. Although all participants owned a VCR, only three reported programming it to record shows more than three times a year.

The VCR and Its Simulation

The HyperCard device simulation was a high fidelity simulation of a Mitsubishi VHS VCR (model HS-3161UR). Programming the VCR was done via its front panel (see Figure 2). The simulation has 18 keys that push on/off, click, or slide. Only eight of these keys are needed to program the VCR. The VCR contains a display area showing time, channel, day-of-week, program number and, if turned on, a program record (PROG REC) indicator. The VCR has three modes: CLOCK SET (default), START, and END. The availability and meaning of the displays varies with the mode.

Procedures

The study had three phases, familiarization, learning, and performance. It lasted approximately 1 hr.

Familiarization. Before the study began, participants were given practice to familiarize them with the location and operation of the 18 keys. For this practice, all labels were removed and all displays were turned off. For each of the 18 keys, a flashing arrow appeared beside it and remained flashing until the key was pushed, toggled, or slid through its full range of positions. Participants went through two rounds of this familiarization procedure. On each round, the keys were cued in a different random order.

Learning. The learning phase used an instructionless learning paradigm. During this phase, participants programmed the same show for five trials. For each trial, participants were told to begin and allowed to work until they believed they had programmed the show or until they gave up. When they indicated they were done with the trial, the experimenter reset the VCR simulation and told the user to program the same show again. This procedure continued for all five trials. During this phase, the experimenter refused to answer any questions regarding the operation of the VCR or the success of the programming efforts.

The motivation for instructionless learning was to enable participants to discover their own rules for interacting with the artifact to perform the task. They were not provided with procedures for programming the commercial VCR.

Performance. The performance phase required participants to program three shows,

TABLE 1
Shows Used in Study (Note that Initial Clock Time was the Actual Clock Time at Which the Participant Participated in the Study. The Initial Channel was Always Channel 9)

show	Name	TIME	DAY	CHANNEL
Reg1	Bob Hope Special	9:00 p.m.–10:30 p.m.	Tomorrow	5
Reg2	Cheers	7:00 a.m.–7:30 a.m.	2 days from now	11
Mid1	Honeymooners reruns	11:30 p.m.–1:00 a.m.	Tomorrow	7
Mid2	Night Court reruns	11:00 p.m.–12:30 a.m.	2 days from now	10

each to the criterion of two consecutive successful trials. This phase began after the fifth learning trial. After new instructions were read, the participants worked on each show until criterion. As in the learning phase, all trials were self-paced. In contrast to the learning phase, after each trial that was not programmed correctly, participants were provided feedback on the first, uncorrected error they had made. Other than this error correction, use of the VCR to program a show was not demonstrated in any phase. As the user programmed the VCR, an action protocol was recorded of all keypresses. Each record indicated the time to the nearest second, what key was pressed, and the state of all VCR registers (e.g., day-of-week, channel, time, and so on). These action protocols constituted the data that were used for model-tracing.

The Shows. Each participant programmed variations on the same four shows, one show during the learning phase and three during the performance phase. The four shows included two regular and two midnight shows. As shown in Table 1, start and end times for regular shows were in the same day, whereas the midnight shows started in late evening and ended in early morning the next day. As the VCR used a 24-hr clock, shows that spanned midnight are slightly more complicated than regular shows. For example, an hour show that begins at 11:30 p.m. and ends at 12:30 a.m. required setting the start time to 23:30 and the end time to 00:30.

Shows Reg1 and Mid1 began the day after the actual day of the experimental session; Reg2 and Mid2 began two days from the experimental session (participants always saw a specific day, for example, *Wednesday*, rather than *tomorrow* or *2 days from now*). As they programmed each show, a 3 × 5 card with the show’s name, time, day-of-week, and channel was placed to the left of the mouse pad, in full view of the participant.

The shows were presented to all participants in one of two orders: Reg1 Mid2 Reg2 Mid1, or Reg2 Mid1 Reg1 Mid2. Hence, during the learning phase, all participants had a regular show, whereas the first show during the performance phase was always a midnight show.

Results and Discussion

This section provides an overview of the empirical results. The focus is on trials to criterion during performance and an analysis that assigns correct and incorrect keypresses to categories in a keypress taxonomy. The next section, *A Least-Effort Perspective on*

TABLE 2
Performance Trials

	s02	s03	s04	s05	s06	s07	s08	s09	s10	Mean	SD
Total trials (shown 2–4)	7	7	6	11	8	8	9	7	7	7.8	1.5
Total correct trials	6	6	6	6	6	7	7	6	6	6.2	0.4
Mean KPs per trial	31.5	28.2	38.0	31.0	32.5	40.1	33.1	31.5	46.2	34.7	5.7
Mean time per trial	57.3	72.3	58.2	59.5	52.3	72.9	61.0	58.7	92.5	65.0	12.4
Time per KP	5.3	4.7	6.3	5.2	5.4	5.7	4.7	5.3	7.7	5.6	0.9

Interactive Behavior, develops a cognitive model of the interactive behavior required to program the VCR. Assumptions incorporated in the model are supported by data from the instructionless learning and performance phases. Finally, in *The Nature, Detection, and Correction of Errors*, model-tracing is used to characterize human behavior as matching or violating the model's behavior. The resultant goal/subgoal taxonomy is analyzed to determine how and why errorful performance deviates from normal performance.

Familiarization and Instructionless Learning

All participants completed two rounds of familiarization. No participant had problems with this task. Considerable learning occurred during instructionless learning. Some of the data from the instructionless learning phase will be presented and discussed in the next section.

Overview of the Data Set

Successfully programmed trials provided the main data set for the study of errors. The criterion for each of the three performance phase shows was two consecutive, successfully programmed trials. Hence, the minimum number of performance trials was six (two trials per show for three shows). Participants completed from 6 to 11 trials each with a mean of 7.8 and a median of 7.0 (see Table 2).

By criterion, each user would have had a minimum of six successful trials. Seven of our nine participants had exactly six; the other two participants had seven. Out of a total data set of 70 trials, 56 trials ended with a correctly programmed VCR and 14 did not.

Time per successful trial ranged from 36 to 143 s with a mean of 65 s and a median of 60 s. Number of keypresses ranged from 22 to 65 with a mean of 34.8 and a median of 31.5. Total number of keypresses for all successful trials for all participants was 1,946.

Keypresses

Each keypress (i.e., mouse click on the VCR simulation—see Note 3) was classified as a hit (kp-hit) or error (kp-error) (see Figure 3). The kp-hit category includes all keypresses that took the VCR one step closer to being correctly programmed. Kp-error applies to

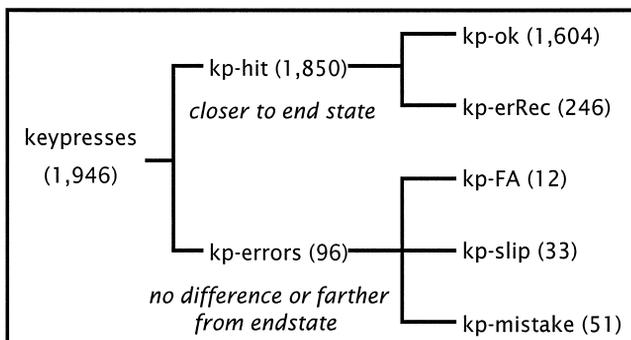


Figure 3. Classification of keypresses. A keypress hit is a keypress that brings the device closer to the end state. A keypress error is a keypress that takes the device farther from the end state or that does nothing.

those keypresses that took the VCR one step or more away from being correctly programmed or that did nothing. Whether a keypress was a kp-hit or kp-error was determined simply and objectively by reference to the behavior of the VCR. The determination was made by comparing the state of the VCR immediately before the keypress with its state immediately after the keypress.

Kp-hits were further classified into kp-ok and kp-error-recovery (kp-erRec). Kp-oks were those keypresses that were in the minimum set required to correctly program the VCR. Kp-erRec were those keypresses needed to recover from prior error. These classifications required knowledge of the minimum set of keypresses as well as the prior history of keypresses.

Kp-errors were classified into three categories: kp-false alarms (kp-FA), kp-slips, and kp-mistakes. Of the 18 keys that could be pressed on the VCR, only eight affected programming; pressing any of the other 11 keys was a kp-FA. Kp-slips were defined by key and by time as *one keypress too many* coming within two seconds of the preceding keypress. All other errors were considered mistakes.

Figure 4 provides an example of the use of the keypress taxonomy. If the VCR is set to Thursday and the target day-of-week for StarTrek is Saturday, then FRI and SAT are kp-hits because each brings the VCR one step closer to the goal state. SUN is a kp-error because it takes the VCR one step away from the goal state; MON TUE WED THR FRI

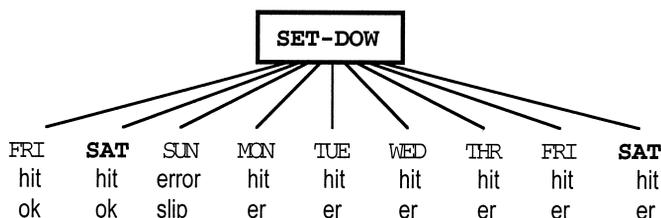


Figure 4. Keypress slip example.

TABLE 3.
Keypress Tally

	s02	s03	s04	s05	s06	s07	s08	s09	s10	Totals	Mean	StDev
KEYPRESS (KP) HIT												
kp-ok	186	150	196	149	159	194	201	177	192	1604	178	20.5
kp-erRec	1	6	28	21	26	62	28	10	64	246	27	22.5
Total Kp-hits	187	156	224	170	185	256	229	187	256	1850	206	36.8
KEYPRESS (KP) ERROR												
kp-FA	0	4	0	0	0	8	0	0	0	12	1	2.8
kp-slip	1	1	2	3	2	8	3	2	11	33	4	3.5
kp-mistake	1	8	2	13	8	9	0	0	10	51	6	4.9
Total Kp-errors	2	13	4	16	10	25	3	2	21	96	11	8.7
TOTAL KEYPRESSES	189	169	228	186	195	281	232	189	277	1946	216	40.9

SAT are all kp-hits, again, because each brings the VCR one step closer to the goal state (this interface does not allow the user to backup; once a slip is made, the only way to recover is by going forward).

At the subcategory level, below kp-hit and kp-error, the first FRI and SAT are kp-oks because these two are among the minimum set of keypresses required by the model. The other kp-hits (MON TUE WED THR FRI SAT) are kp-erRecs (indicated in Figure 4 as “er”). These are needed only because the participant has made an error that put the VCR into some state from which extra keypresses are required to get to the goal state. The kp-error SUN is a kp-slip because it was “one keypress too many” coming within two seconds of the preceding keypress.

Table 3 provides a detailed breakdown of the keypress data by user. The 56 trials provided 1946 keypresses that were divided into 1850 kp-hits and 96 errors. The kp-hits were further divided into 1604 kp-oks and 246 kp-erRec keypresses. The 1,604 kp-oks were the minimum number of keypresses needed to correctly program the 56 shows. The remaining 246 kp-hits were kp-erRecs.

Although the 51 kp-mistakes and 33 kp-slips (see Table 3) seem rare when compared to the total data set of 1946 keypresses, they are important. Whereas only 4% of the keypresses made while programming the VCR were mistakes or slips, if any one of these keypresses had not been detected and corrected, the VCR would not have been successfully programmed. Problems with just 4% of the keypresses would have prevented shows on 37 of the 56 trials (66%) from being successfully recorded.

III. A LEAST-EFFORT PERSPECTIVE ON INTERACTIVE BEHAVIOR

Card et al. (1983) suggested a steep prerequisite for the study of errors in routine, rule-based tasks; namely, that such a study must rely on a fairly complete account of the cognition required to perform the task. We must have confidence that the control structure used by the analyst to classify errors is the control structure used by participants to perform the task.

Confidence in the control structure is increased if it yields an account of error-free interactive behavior, and if overt errors can only be produced by violating the control structure. However, violations of the control structure that do not result in overt errors are a gray area. Do such violations represent true errors or do they signal a failure of the model's control structure to capture the user's control structure?

It is too much to expect any model to trace all correct behaviors by all participants. However, if behavior that the model cannot trace is to be interpreted as an event that requires explanation, then confidence that the model can trace the vast majority of correct behavior must be very high.

The model proposed in this section is based on three types of analyses. First is a simple task analysis of the correct behavior required to videotape a show using the commercial VCR. Second is an analysis of interactive behavior in the instructionless learning phase of the empirical study. Third is an analysis of interactive behavior during the performance phase of the empirical study. These analyses of interactive behavior were guided by three cognitive engineering principles derived from a least-effort perspective on cognitive performance.

In this section, I briefly review studies that support the inference that the cognitive system operates according to a least-effort principle. The review is followed by an introduction to three least-effort principles of cognitive engineering. Each principle is discussed in some detail along with data that show its influence on the interactive behavior observed in this task. Illustrative implications of each principle for the proposed model are presented.

The claim advanced in this paper is that for trials that end successfully, the proposed model captures the control structure used by participants in programming the VCR. Hence, the model should provide insights into violations of this control structure as well as their detection and correction. These insights are discussed in the penultimate section of this paper, *The Nature, Detection, and Correction of Errors*.

A model that can provide an account of errors should account for most correct behavior. Although accounting for successful behavior cannot be considered a strong test of the model, it is a necessary test. This test is presented in the *Accounting for Success* part of the current section. Finally, some trials in the performance phase ended unsuccessfully. These unsuccessful trials provide an interesting means of evaluating the model. If the participants have the control structure hypothesized by the model, they will detect and correct any errors they might make. Hence, if the model does capture correct performance, errors it does not detect and correct must arise from factors that lie *Outside the Model* (these are discussed below).

Cognitive Least-Effort

The interactive behavior required to perform a rule-based task emerges through a combination of constraints and opportunities provided by the interaction of embodied cognition with a task and an artifact designed to accomplish the task—the embodied cognition, task, and artifact triad (see Figure 5). The least-effort principle asserts that, all else being

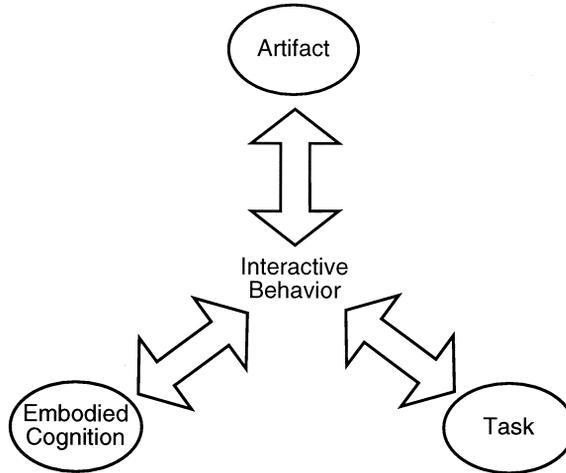


Figure 5. The interactive behavior of an operator using an artifact to perform a task arises from the limits, mutual constraints, and interactions between and among each member of the embodied cognition, task, and artifact triad. For the commercial VCR, this yielded a particular task-to-device rule hierarchy and a set of procedures for place-keeping (see text).

equal, the interactive behavior that users adopt arises from an implicit attempt to minimize costs while maximizing benefits.

In the world of economics it is recognized that real-world optimization is impossible and that “the real economic actor is in fact a satisficer” (see, e.g., Chap. 2, Simon, 1996). More recently, J. Payne, Bettman, and Johnson (1993) applied the cognitive least-effort principle to a variety of decision-making paradigms. Similarly, Anderson has used a rational analysis approach to derive least-cost predictions concerning, among other things, causal inference and problem solving (Anderson, 1990; Anderson, 1991; Anderson & Schooler, 1991).

The approach adopted here was to extend the least-cost approach to explain mundane interactive behavior. The slogan is *milliseconds matter*. The implication is that costs on the order of a few hundred milliseconds suffice to influence interactive behavior. The interactive behavior adopted, for the most part, will not result from a conscious and deliberate consideration of costs and benefits, but will emerge gradually through routine interactions with the device.⁵

Least-effort in interactive behavior relies for its support on a combination of intuition and research. Intuitively, it makes sense that, all else being equal, once a user starts to press, say the hour button, he or she will continue pressing that button until the desired hour has been reached. Likewise, once the user begins to set the show’s start time, he or she will complete the task by setting all elements of the time (i.e., using the HOUR, 10MIN, and MIN key as needed—see Figure 2) before setting, for example, the show’s channel or day-of-week.

Congruent with these intuitions, research shows a time cost in shifting attention from one task to another. For example, consider two simple classification tasks: classifying

numbers as odd-even and classifying letters as vowel-consonant. In tasks that involve alternating runs of numbers versus letters, the first trial of the new run (switching from number to letter or from letter to number) takes approximately 150 ms longer than the time required for subsequent trials. This ubiquitous finding holds over many variations (e.g., see Altmann & Gray, 1998; Altmann & Gray, 1999a; Rogers & Monsell, 1995).

Time costs in the range of 150 ms are undoubtedly below the level of conscious awareness; however, research shows that the strategies that people adopt are sensitive to costs of this order of magnitude. For example, changing the effort involved in information gathering from one requiring an eye movement to one requiring a mouse movement influenced the decision making strategies adopted by participants in a classic decision-making paradigm (Lohse & Johnson, 1996). A still smaller change, from an eye movement to a head movement, induced a shift in a block-copying paradigm from a *memoryless* strategy to a memory intensive strategy (Ballard, Hayhoe, & Pelz, 1995). Similarly, when the cost of making a move in the 8-puzzle problem increased from one keypress to several, the strategy used to solve the puzzle shifted from one in which search was “reactive and display-based” to one in which search was more plan-based (O’Hara & Payne, 1998). In all three examples, small differences in the effort required sufficed to affect the strategies adopted.

Three Least-Effort Principles of Cognitive Engineering

The model of interactive behavior adopted in this paper is composed of the task-to-device rule hierarchy shown in Figure 6 and a particular rule for place-keeping. The claim is that this control structure emerges when the interactions of embodied cognition with task and artifact are governed by the general principle of cognitive least-effort. Specifically, the model of interactive behavior is shaped by three least-effort principles of cognitive engineering.

- Least-effort in operating the device
- Least-effort in mapping prior knowledge to device knowledge
- Least-effort in place-keeping

Each principle is elaborated below.

Artifact Influence on the Hierarchy: Least-Effort in Operating the Device

The artifact influences the task-to-device rule hierarchy through the principle of least-effort in operating the device. For example, implicit in Figure 6 is the assumption that once the user begins to press, for example, the hour button, he or she will continue pressing it until the hour is set. This claim was supported by the data. Across the 56 successfully programmed shows, 599 keys (excluding the occasional KP-FA, see Figure 3) were pressed one or more times during 634 episodes of keypressing (where a keypress episode was one or more contiguous clicks of the same button). The number of episodes

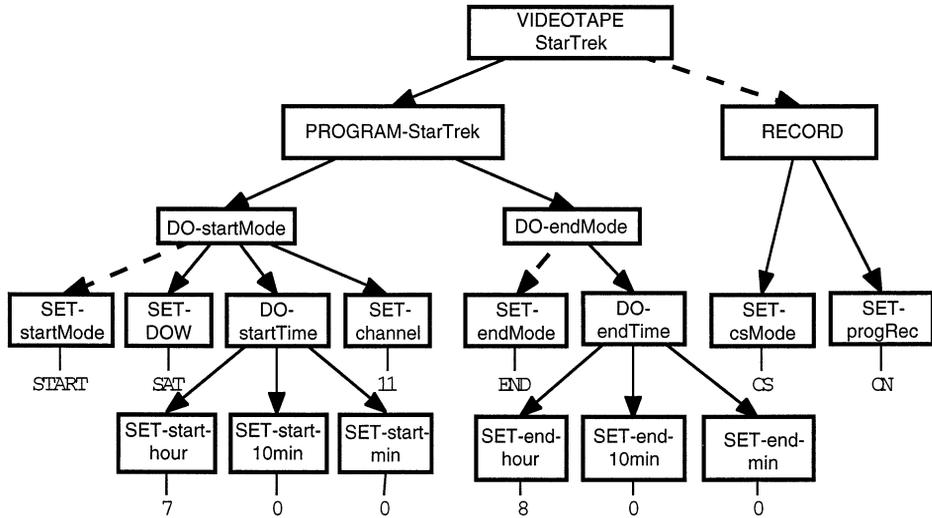


Figure 6. A mapping of the task goals from Figure 1 onto the device shown in Figure 2. This task-to-device rule hierarchy is largely determined by soft constraints. (Subgoals are represented by boxed nodes. Leaf nodes are unboxed and may represent multiple keystrokes. The dashed line leading from DO-startMode and DO-endMode indicate that subgoals SET-startMode and SET-endMode must be performed before the others. Contrariwise, the dashed line from VIDEOTAPE to RECORD indicates that RECORD must be performed last. With those three exceptions, the subgoals of a goal may be performed in any order.)

per key per show was 1.058. The small number of revisits (i.e., 0.058 per episode) supports the claim that the nodes in Figure 6 that govern keypresses (the various SET-<goals>, e.g., SET-start-hr and SET-DOW) are goals in this task. The 35 episodes (i.e., 634–599) in which the same key was visited twice in one show violate the least-effort principle by requiring two additional switches in serial attention with concomitant finger movements to and from another key. Hence, these 35 revisits are violations of the goal hierarchy that need to be explained.

Predictions become more interesting when the design of the device imposes a least-effort structure that is not a necessary part of the task-based goal hierarchy. For example, the hierarchy shown in Figure 1 makes programming the day-of-week, start time, channel, and end time siblings of the same parent node. In contrast, the task-to-device rule hierarchy (see Figure 6) places the first three under DO-startMode and end time under DO-endMode.

This difference between Figure 1 and Figure 6 implies two claims. First, that DO-startMode and DO-endMode are imposed by the device. Second, that as users gained experience with the commercial VCR they treated all settings that could be made in a given mode as part of the same subgoal.

The first claim was supported by keypress data gathered during the 45 instructionless learning trials (9 participants \times 5 trials each). During instructionless learning, modes were rarely used correctly and, as the keypress data show, often not used at all. Indeed, out of

45 instructionless learning trials, the mode switch was not touched on seven trials, it was set once and then ignored on 12 trials, and it was set twice and then ignored on 13 trials. Only on 13 trials out of 45 was it used to toggle between all three modes (START, END, and CLOCK SET). In contrast, all keys required to implement the four task-based goals were pressed by all but one participant, on all but one trial. When compared to the spontaneous use of keys that directly implemented the task-based goals, the nonuse and partial-use of the mode switch suggests that despite the labels (see left side of Figure 2) participants did not initially think of the videotaping task as one that divided itself into a series of mode-specific subgoals.

The second claim was supported by data gathered during the performance phase. Across all 56 successfully programmed shows, one participant, on one trial, switched from start mode to end mode to start mode. On all other trials (55/56), all participants visited start mode once and end mode once. The small physical cost imposed by the mode switch sufficed to induce participants to adopt the DO-startMode and DO-endMode goals shown in Figure 6.

Influence of Prior Knowledge on the Hierarchy: Least-Effort in Mapping Prior Knowledge to Device Knowledge

Prior knowledge influences the task-to-device rule hierarchy through the principle of least-effort in mapping prior knowledge to device knowledge. The claim is that, with all else equal, it is easier to reuse a cognitive structure (e.g., a chunk of knowledge) than it is to create a new one or to ignore an existing one. The best illustration of this principle in the commercial VCR is in programming the time.

The knowledge-influenced claim is that people regarded SET-start-hour, SET-start-10min, and SET-start-min as three subgoals of DO-startTime and SET-end-hour, SET-end-10min, and SET-end-min as subgoals of DO-endTime. As switching attention from one goal to another entails a switch cost, once, for example, DO-startTime is begun it should be completed (i.e., its three subgoals should be pushed and popped) before moving to another goal. Note that the existence of DO-startTime is not enforced by the design of the artifact. From the perspective of least-cost in operating the device, it is no more effortful to go from, for example, 10MIN to DAY OF WEEK than it is to go from MIN to HOUR.

The above analysis implies that interleaving the three DO-startTime subgoals with a fourth goal is an error. For example, the sequence SET-start-hour, SET-start-min, SET-start-10min, SET-DOW is correct but the sequence SET-start-hour, SET-DOW, SET-start-min, SET-start-10min is an error. From a least-cost perspective, the second sequence incurs an additional cost to switch attention back to DO-startTime after completing SET-DOW.

To evaluate this claim we can look at the number of shows for which all required time settings were made contiguously in the performance phase of the empirical study. For DO-startTime, in 55 of the 56 successfully programmed trials (98.2%) time settings were made contiguously; that is, without intrusions. For comparison, the second highest

contiguous setting was between SET-DOW and SET-start-hour. The transition SET-DOW → SET-start-hour occurred 80% of the time with SET-start-hour → SET-DOW occurring 3% of the time for a total of 83% contiguous settings. The third highest contiguous setting was between SET-channel and SET-start-min. The transition SET-start-min → SET-channel occurred 72% of the time, while the transition SET-channel → SET-start-min never occurred.

For DO-endTime the equivalent comparison is not valid in that the contiguous setting of SET-end-hour, SET-end-10min, and SET-end-min does not discriminate between DO-endTime and DO-endMode. For both cases, SET-endMode must be performed before any of the time settings.

Device-Influenced versus Knowledge-Influenced

In an easy to use device, least-cost considerations of operating the device would work with least-cost considerations of mapping prior knowledge to device knowledge. The commercial VCR demonstrates this cooperation in several ways.

Of the five keys (DAY OF WEEK, HOUR, 10MIN, MIN, and up or down CHANNEL—see Figure 2) needed to implement the four task-based goals (see Figure 1), all were pressed on 44 of the 45 instructionless learning trials. (One key was not pressed by one participant on one trial.) Apparently, a general label-following strategy (Polson & Lewis, 1990) sufficed to guide participants to the correct keys.

Label-following involves more than simply label reading. Indeed, of the 10 keys on the commercial VCR that never had to be pressed, a mean of 3.8 were pressed on the first instructionless learning trial. This number declined to 1.0 by the fifth instructionless learning trial. The CE+ model (Polson & Lewis, 1990) implies that the decline in spurious keypresses during instructionless learning results from a causal analysis of actions and system responses. For the commercial VCR, this analysis sufficed to focus participants on the correct set of keys.

The above example illustrates a case in which the design of the artifact worked with task-based knowledge to support the emergence of interactive behavior. Unfortunately, the two are not always mutually supportive. For example, the second level of the task-to-device rule hierarchy (see Figure 6) contains two goals: PROGRAM-StarTrek and RECORD. The first, PROGRAM-StarTrek, incorporates all task-based goals as well as several device-specific goals. The second, RECORD, is a purely device-specific goal that involves returning the VCR to the CLOCK SET mode and turning on PROG REC after the show has been programmed.

From the perspective provided by Byrne and Bovair (1997) setting the VCR to RECORD is a postcompletion action that is susceptible to postcompletion error. During the performance phase, 14 trials ended in error (i.e., a VCR that would not have recorded the target show). Of these 14 trials, six of them included a postcompletion error; namely, not setting the VCR to RECORD.

A more direct conflict between the design of the artifact and pre-existing knowledge concerned setting the time. The commercial VCR used a 24-hr clock. During the

performance phase, 8 of the 14 trials that ended in error (i.e., a VCR that would not have recorded the target show) involved some problem with the 24-hr clock (e.g., setting start time to 7 a.m. rather than to 19 or to 12:30 p.m. rather than to 00:30 a.m.).

In addition, dividing the setting of time into three subgoals (e.g., SET-start-hour, SET-start-10min, and SET-start-min) seems to have introduced an enduring conflict between prior knowledge and device-specific knowledge. For those shows that were completed, a large number of errors (that were later detected and corrected) concerned the 10min setting. The 10min display required the participants to count 1 – 2 – 3 – 4 – 5 – 0. As discussed below, an interesting class of errors can be explained by postulating that the design of the commercial VCR violated cultural expectations that “6” will follow “5” and that “0” will be preceded by “9.”

Display-Based Difference-Reduction: A Strategy of Least-Effort in Place-Keeping

Place-keeping entails knowing what parts of the task have been completed and what parts remain to be accomplished. A nightmare design might require the user to keep the entire task-to-device rule hierarchy (see Figure 6) in his or her head, traversing the nodes in a depth-first fashion, mentally marking the most recently visited node as completed as soon as the next node is begun. Furthermore, when a SET-<goal> (e.g., SET-start-hour) is begun, such a design might require the user to mentally calculate the difference between the current hour and the target hour, and to subtract one from this difference each time the HOUR button is pressed.

This nightmare design illustrates both the global and local aspects of place-keeping. Global place-keeping keeps track of what goals have been accomplished and what goals remain to be accomplished. Local place-keeping keeps track of progress on the current goal.

For the commercial VCR, the cognitive burden of place-keeping is largely alleviated through display-based difference-reduction. In difference-reduction (also called hill-climbing) (for an introduction, see Anderson, 1995; for detailed examples, see Newell & Simon, 1972) the user proceeds by gradually reducing differences between the current state of the world and the goal state. As each step makes the current state more similar to the goal state, past states do not have to be retained and planning more than the next step is not required.

Difference-reduction is an optional, not a necessary, component of display-based systems. For example, in making coffee, display-based difference-reduction performs local place-keeping by indicating how much water to put in the reservoir; simply, “pour water until filled.” Other steps of coffee-making with display-based components may not involve difference-reduction but, instead, may require the retention of intermediary states. For example, the number of scoops of coffee required to make a pot may involve formulating a subgoal such as “put six rounded scoops of coffee into the filter.” Unlike the reservoir-filling example, successful execution of this subgoal may involve keeping track of intermediary states; that is, how many scoops have been put in and how many more must be added. Likewise, if the coffee maker were interrupted in the midst of adding

coffee to the filter, on her return to this task, its display-based features would suffice to let her know that she had begun this task, but would not suffice to allow her to determine whether, for example, three or four scoops had been added.

As in the task of making coffee, the commercial VCR partially implements display-based difference-reduction. For global place-keeping, the user has to remember the task-based and device-specific goals. However, the user does not have to remember the state of those goals. For example, a glance at Figure 2 indicates the following; the VCR is in start mode (indicated by the position of the mode switch and the label “on” in the left side of the display), the day-of-week setting is SAT (correct for StarTrek), the channel setting of 9 is different from the target setting of 11 (and needs to be set), the start time of 19:01 is different from the target start time of 19:00 (and needs to be set). Focusing on start time reveals that the start hour is correct for recording StarTrek (indicated by the 19), the start ten minute setting is correct (indicated by the 0), but the start minute setting of 1 is different from the target setting of 0 (and needs to be set).⁶ As the end time settings cannot be checked from the START mode, the user must remember that end time needs to be completed. Hence, in contrast to the nightmare scenario pictured above, for the commercial VCR most (but not all) global place-keeping can be managed by global display-based difference-reduction.

Display-based difference-reduction transforms local place-keeping from a predominately cognitive to a largely perceptual task. For example, each click of the HOUR button increments the hour display by one. Users must only keep in mind the target hour and decide when the hour on the display matches the target hour.

The least-effort arguments for the use of display-based difference-reduction are similar to those for the proposed task-to-device rule hierarchy. Difference-reduction is a strategy that is available to the participants and is supported by the artifact. Its use requires less cognitive cost than the alternatives. Difference-reduction involves the firing of fewer rules as well as the creation and maintenance of fewer chunks in declarative memory.

An interesting implication of this analysis is that, for the current model, the detection and correction of errors should be largely *business-as-usual*. The task-to-device rule hierarchy and the display-based difference-reduction that are postulated as controlling correct behavior should suffice to detect and correct errors. For the VCR, detection and correction of most errors will be routine.

Summary

The three least-effort principles of cognitive engineering were applied to the analysis of the interactive behavior required to program the commercial VCR. The goal of this effort was a model of error-free performance; one that would be capable of tracing the vast majority of each participant’s error-free keypresses. The resultant model has two characteristics that need to be emphasized. First, the proposed task-to-device rule hierarchy minimizes the cognitive and motor costs of using the device. Violations of the task-to-device rule hierarchy that do not result in overt errors are non-overt errors that need to be explained. Second, display-based difference-reduction provides a *business-as-usual* ac-

count of error detection and correction. Errors in programming this particular VCR do not throw the device into some unique *error state*. Rather, errors produce a non-target start time, end time, channel, or day-of-week. The procedures required to recover from these non-target settings are the same procedures required to achieve the target settings (e.g., StarTrek, 7 to 8 p.m., Saturday, channel 11). Although such a business-as-usual model of error correction is not expected to work for all devices, as the data will show, it works surprisingly well for this device.

Accounting for Interactive Behavior

By itself, the current model will correctly program any show for which it is given time, day, and channel information. This demonstrates that the model is complete enough to perform the task. However, for the protocol analysis, the model's role is to trace an individual's error-free performance; this involves one more set of steps (see e.g., Anderson, 1993; Anderson, Boyle, Corbett, & Lewis, 1990). There are several places during programming where the user has options as to what goal to work on next. For example, while in DO-startMode (see Figure 6) after setting the mode to START, the user may choose to program either the day-of-week, start time, or channel. To the model, such goals appear as equally attractive next steps. In the absence of experimenter intervention, the model would randomly choose one of these steps. In contrast, when model-tracing, the experimenter stops the model at such points and resolves the conflict by picking the goal that the user picked. This resetting suffices to account for almost all error-free human performance (the exceptions are discussed later).

Relationship to ACT-R

ACT-R (Anderson & Lebière, 1998) is a candidate architecture of cognition that is partially implemented as a programmable modeling language. ACT-R the theory is composed of many richly detailed cognitive mechanisms. ACT-R the modeling language was designed to build models that incorporate individual cognitive mechanisms without the necessary requirement that each model include all mechanisms.

In the work reported here, the models built made use of the most basic of ACT-R's mechanisms. These mechanisms—ACT-R's goal hierarchy, its conflict resolution, and its distinction between declarative and procedural memory—were little more than those that would be contained in a generic theory of information processing such as GOMS (Card et al., 1983). Indeed, the major advantage of ACT-R over GOMS was that ACT-R is runnable and therefore easier to trace than a comparable GOMS model would be. This advantage was especially important early in the modeling effort when several alternative models were built and used to directly interact with the simulated VCR.

These models were built in ACT-R 2.0 (Anderson, 1993) and used AppleEvents™ to communicate with a simulated VCR built in HyperCard™. The Lisp-based ACT-R models were able to directly interact with the same computer-based visual display (i.e., the VCR device simulation) that human participants used. The model used in model-tracing

is a hybrid of those earlier models that, as discussed above, incorporates both display-based difference-reduction and the task-to-device rule hierarchy shown in Figure 6. All of the models explored were expert models—all correctly programmed the VCR without error.

In contrast to ACT-R the modeling language, wider use was made of ACT-R the theory. In particular, ACT-R's rational analysis of memory inspired the least-effort approach that resulted in the discovery of display-based difference-reduction and the task-to-device rule hierarchy.

Accounting for Success: Matches to the Goal Structure

If the model were given the same exact conditions and tasks as the participants and allowed to run without experimenter intervention (i.e., not model-tracing), across the 56 shows it would generate 984 goal pushes (or 984 goals and subgoals set) and an equivalent number of goal pops (i.e., goals and subgoals completed). For model-tracing, the task to device rule hierarchy shown in Figure 6 was used to guide a walkthrough of the participants' action protocols. The data discussed here are the counts of pushes and pops needed for the model to trace each participant's correct keypresses.

If the goal structure of the model is the goal structure adopted by the participants, model-tracing should generate most of the 984 goal pushes and pops. The walkthrough came very close to this mark, generating 977 of the 984 predicted goal pushes. These goals are classified in Table 4 and shown in Figure 7 as psh-match-oks. The seven missing psh-match-oks are all from one participant. This participant had a slightly different task-to-device rule hierarchy than that predicted by the model. As these trials ended correctly, the correct behavior was performed; however, it was not performed at a place in the task-to-device rule hierarchy predicted by the model. Hence, the model classifies these seven missing psh-match-oks as psh-violates. (The discussion of the seven missing goals is elaborated later.) Once one of the 984 goals is pushed, even if it is pushed in the wrong place, its successful completion is scored as a pop-match-ok. As Table 4 shows, all 984 pops were captured by the model and scored as pop-match-oks. The success of the model at tracing 977 of the 984 required goals (99.3%) indicates that the device-influenced task-to-device rule hierarchy used by the model was very similar to that adopted by the participants.

Outside the Model: Undetected Errors

The nature of errors undetected by participants is important to the claim that the hypothesized control structure is similar to that used by participants. Once acquired, the control structure should enable participants to detect and correct certain categories of errors. Errors outside of these categories may indicate problems with acquiring or using the control structure. However, if the typical undetected error were one that the model should detect but did not, some account as to why the model failed would be necessary. On the other hand, if the typical undetected error were one that the model could *not* detect,

TABLE 4.
Goal Push/Pop Taxonomy Data Set Summary

	s02	s03	s04	s05	s06	s07	s08	s09	s10	Totals	Mean	SD
GOAL or SUBGOAL PUSH												
PUSH-MATCH												
psh-match-ok	106	106	106	105	106	122	115	107	104	977	108.6	6.0
psh-match-recovery	10	1	2	1	1	11	0	0	4	30	3.3	4.2
Total Push Match	116	107	108	106	107	133	115	107	108	1007	111.9	8.7
Total Psh-Violate	1	7	3	15	5	10	7	0	4	52	5.8	4.7
TOTAL GOAL PUSH	117	114	111	121	112	143	122	107	112	1059	117.7	10.7
GOAL OR SUBGOAL POP												
POP-MATCH												
pop-match-ok	106	106	106	105	106	122	122	107	104	984	109.3	7.2
pop-match-recovery	0	0	1	0	1	5	0	0	2	9	1.0	1.7
pop-misc	2	7	4	16	5	10	0	0	4	48	5.3	5.1
Total Pop-Match	108	113	111	121	112	137	122	107	110	1041	115.7	9.6
POP-VIOLATE												
pop-violate-premature	9	1	0	0	0	6	0	0	2	18	2.0	3.3
pop-violate-postponed	1	0	2	3	2	7	3	2	10	30	3.3	3.2
Total Pop-Violate	10	1	2	3	2	13	3	2	12	48	5.3	4.8
TOTAL GOAL POP	118	114	113	124	114	150	125	109	122	1089	121.0	12.2

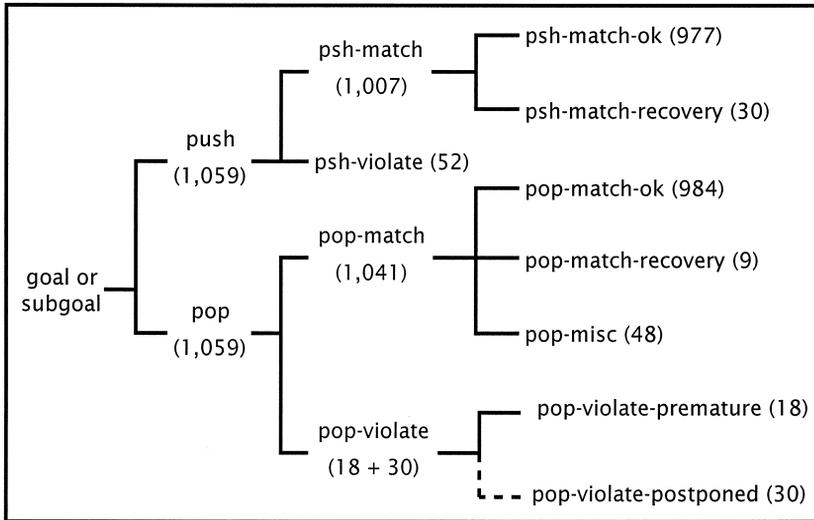


Figure 7. Goal push/pop taxonomy is used to categorize successes and failures of the control structure. Unlike keypresses, individual pushes and pops of goals in the goal hierarchy are not judged by whether they bring the device closer to or farther from the end. Rather, they are judged by whether they match or violate the control structure. Note that the pop-violate-postponed category represents instances in which to trace behavior a goal had to be stopped from popping. Hence, the 30 instances of this category are in addition to the 1059 pops traced by the model.

this would strengthen the argument that, in their successfully programmed trials, participants used the hypothesized control structure.

The nine participants had a total of 70 trials (excluding five learning trials per participant). Of these 70 trials, 56 ended in a correctly programmed VCR, the other 14 trials ended in error (a VCR that would not have recorded the target show). A total of 26 terminal errors were made on these 14 trials; that is, at the time the participant said the trial was completed there was one or more errors that would prevent the VCR from correctly recording the show.

The undetected errors can be placed into four categories.

24-hr Clock. Many errors were due to the participant's unfamiliarity with the 24-hr clock. For example, the end time of a show that went from 11 p.m. to 12:30 a.m. was set, not to 00:30 (AM) but to 12:30 (PM). Another show that began at 7 p.m. was set, not to 19:00 (PM), but to 07:00 (AM).

Mode Errors. These primarily affected time. Some of these include setting the start time correctly, but then programming the end time over it; that is, without going into endMode. Another such error was to set the clock time rather than start time (hence NOW would be set to the start time of the show).

Postcompletion Errors (Byrne & Bovair, 1997). After the show was correctly programmed, the last step was to set the VCR to CLOCK SET mode and to press the PROG

TABLE 5
Undetected Errors

Failures	24 hr	Mode	Post-completion	Should-haves	Total
Prior to first okay trial	5	5	6	3	19
After first okay trial	3	1	0	3	7
Total	8	6	6	6	26

REC button. As indicated by Figure 6, the subgoals and actions associated with RECORD are prime candidates for postcompletion error.

Should-haves. Should-haves are errors that the model should have detected. These involved incorrectly set channel, day-of-week, start time, or end time (other than mode or 24 hr clock errors).

Of these four types of errors, 24-hr clock and mode errors are knowledge errors and would not be caught by the current model; that is, knowledge of the 24-hr clock and the moded interface was built into the model’s task-to-device rule hierarchy. Postcompletion errors are very interesting; however, the model on which this paper is based had no working memory limitations and, hence, could not suffer losses due to working memory.⁷

By category (see bottom row of Table 5) there were 20 errors that the current, business-as-usual, model could not detect and six that it could. Of the six should-have errors, three occurred in conjunction with one or more other errors. The other three were the sole error and if they had been caught by the participant, the show would have been correctly programmed. Hence, the typical undetected error is one that the current model could not detect.

The undetected errors can be further analyzed to shed light on the learning process. Of the 14 failure trials, nine occurred before the first correctly programmed trial. By definition, participants received no feedback during instructionless learning. The first time their errors were pointed out to them and the correct response provided was during the performance phase. Hence, seven participants (out of nine) incorrectly programmed the first trial of the first show of the performance phase. One of these participants incorrectly programmed the first three trials of that first show (for a total of nine failure trials before the first correct show).

If the trials before the first okay are thought of as occurring before the participants’ learning the task-to-device rule hierarchy, it makes sense to look at errors separately by *prior to first okay trial* versus *after first okay trial*. This breakout is provided in Table 5.

The overwhelming number of errors made before the first correctly programmed trial (16/19) were errors that the business-as-usual model could not detect. This outcome supports the contention that the rules contained in the model are similar to the rules used by the participants on their successfully programmed trials. (Note that only one of the three should-have errors occurred by itself, the other two occurred on trials with multiple other errors.)

Errors after the first okay trial, occurred (by definition) after we have some evidence that the rule-set had been learned. Across these five errorfult trials, seven errors occurred,

three of which should have been caught by the business-as-usual model. Two of these three were the sole error on the trial. The third was a time related error (involving the start time) that occurred on the same trial as a 24-hr error occurred for end time. (This participant had demonstrated prior problems with the 24-hr clock.) It could be argued that this should-have error is related to a 24-hr error. Hence, the tally would become two should-haves versus five others (rather than three versus four as shown in Table 5).

The number of should-have errors is small and, even when judged by the most conservative tally (that which separates errors before the first okay trial from errors after the first okay trial), they are a minority. Indeed, looking at the data from another perspective, these errors form a very small part of the performance phase of the study. Fifty-six trials were successfully programmed. If *after* the first okay trial, all should-have errors were caught, the number of successful trials would have gone from 56 to 58.

Summary: A Model of Interactive Behavior

If the sole goal of the current paper was to classify keypresses in an objective manner then, arguably, this goal was met by the keypress taxonomy shown in Figure 3 and Table 3. However, the goal is more complex; namely, to provide a cognitive framework for discussing the nature, detection, and correction of errors. The keystone of this analysis effort is the model of interactive behavior required to program the commercial VCR. The model is based on a least-cost analysis of the constraints and opportunities provided by the embodied cognition, task, and artifact triad.

To program the same 56 shows that the participants programmed, the current model would generate 984 goals and subgoals. For model-tracing, the model was used to walkthrough each participant's keypresses. In this manner, 977 of the predicted 984 goals were generated. These 977 goals represent a 99.3% successful match between the model's predictions and the behavior obtained. Likewise, performance trials that ended unsuccessfully, were unsuccessful for reasons that lie largely outside the current model.

The highly successful match of correct behavior to the claims and predictions of the current model supports the conclusion that the control structure of the model mirrors the control structure used by participants. Given the large number of matches to the control structure, violations achieve the status of phenomena that need to be explained. Likewise, the success of the model at accounting for correct pushes and pops suggests that the goal push/pop taxonomy can be interrogated to yield accounts of the nature, detection, and correction of errors.

IV. THE NATURE, DETECTION, AND CORRECTION OF ERRORS

Earlier, model-tracing was used to determine how well the model accounted for correct behavior. In this section, data are presented from using the model to trace all 1946 keypresses. Steps that the model would make are *matches* to the model. Steps that the model would not make are *violations* of the model. Tracing the 1946 keypresses identified 1059 instances where a goal or subgoal was pushed and 1059 instances where a goal or

subgoal was popped. These pushes and pops form the data set for the *goal push/pop taxonomy* (see Table 4 and Figure 7).

As shown in Figure 7 there are two ways for a goal to match the goal structure—psh-match and pop-match—and two ways for a goal to violate it—psh-violate and pop-violate. Hence, the current analytic framework makes a fundamental distinction between push errors and pop errors.

By definition, the analyst identifies an error when normal model-tracing does not suffice to follow a participant's action protocol. If the new goal is in the task-to-device rule hierarchy (see Figure 6) then the analyst must push and pop goals to make the model follow the participant. When the new and old goal are both within the task-to-device rule hierarchy, the nature of the error can be characterized in terms of the number of links traversed and the relationship between the old and new goal. When an error was detected, we can ask the same question; namely, how many links and what relationship was there between the goal at which detection occurred and the error. We can further ask, what steps were needed to correct the error.

The Nature of Errors

Push Errors

Using the model to trace human performance yielded 52 push violations. Examining these violations from the perspective of the model and memory theory suggested three bases for their origin: task-to-device rule hierarchy failures, display-induced errors, and violations due to either misencoding or misretrieval.

Task-to-Device Rule Hierarchy Failures. Nineteen push violations involve the PROG REC key. After the show has been programmed, this key (see Figure 2) must be pushed in for the show to actually be recorded. Although a show may be correctly programmed, until PROG REC is set (after which it is impossible to watch other tapes or television) the recording will not actually take place.

Whereas PROG REC can be set at any time during programming, the model will only set it after the entire show has been entered. The seven missing psh-match-oks (discussed above) came from s08 who consistently turned PROG REC on after completing the startMode but before starting the endMode (see Figure 6); these seven goals are classified as psh-violates. Twelve related psh-violates came from another user, s05. Throughout all six correct trials, s05 would turn on PROG REC immediately after entering startMode, program the start time, day-of-week, and channel (not necessarily in that order), *turn off PROG REC*, enter endMode, program the end time, enter CLOCK SET mode, and *turn on PROG REC again*.

Although s05's behavior represents extra work (and therefore violates the least-effort principles represented in the model), it does result in a successfully programmed show. It also indicates two mismatches with the model's task-to-device rule hierarchy. For s05, not only is each initial PROG REC classified as a psh-violate, but the six *undings* of PROG

REC before entering endMode, are scored as psh-violates (once PROG REC is turned on, the model will not turn it off).

The episodes with the PROG REC goal are the only ones for which there is clear evidence that the task-to-device rule hierarchy adopted by some participants differed from that built into the model. These are the only correct and consistent behaviors that violate the model's task-to-device rule hierarchy. The fact that all other consistent and correct behaviors can be traced by the model supports the assertion that the model's task-to-device rule hierarchy is very similar to that adopted by the participants.

Display-Induced Errors. Two types of push violations seem to have been induced by the display. The first was made by two different participants (one instance for each participant). In both cases, the participants were in endMode and realized that they had not correctly set the channel. (More on the detection of errors below.) Although the channel display is visible in endMode, pressing the CHANNEL buttons in this mode has no effect. To get the model to trace this behavior requires popping out of endMode, pushing startMode, and pushing SET-channel. (SET-channel is three links from DO-endMode and can be considered its nephew.)

The second push violation suggests a complementary display-induced error. For this error, each of five participants pressed the DAY OF WEEK button while in endMode. (In common with SET-channel, SET-DOW is three-links away and a nephew of DO-endMode.) For the commercial VCR, when endMode is entered and initialized the display shows the time and channel that were set in startMode. The day-of-week is not displayed. In each case, the show was a midnight show in which the day-of-week for the end time would be different from the day-of-week for the start time. For one participant, the psh-violate was made on the first trial of her first midnight show. However, each of the other four participants had one or more trials on a midnight show before making this error. (Indeed, for one participant, this psh-violate was made on the second trial of her second midnight show.) The scarcity of this error (each of five participants made it once) combined with its regularity (it was made only on midnight shows) suggests a curious interaction of device-knowledge and show knowledge with the predominantly display-based nature of the VCR.

These display-induced errors are similar to, but different from, those observed by Larkin (1989). In her coffee pot example users made errors when an internal state of a predominantly display-based task was invisible. An example of this type of error is pouring water from a carafe into an already filled reservoir when the reservoir was an opaque container. Larkin's example differs from the current case in that here a predominantly display-based interface *lured* participants into making an error. In one case, seeing a wrong setting (the channel) suggested to participants that they should be able to fix it. In the other case, not seeing the day-of-week in the endMode, under circumstances in which it was not clear whether the day-of-week needed to be set (i.e., a midnight show), caused half of the participants to try once to set it.

These complementary errors appear to have a common origin. In both cases, there is a conflict between the principle of place-keeping—display-based difference-reduction—

and the rule hierarchy imposed by the principle of least-effort in operating the device. Display-based difference-reduction suggests that there is a goal that needs to be accomplished, whereas the design of the device prohibits the goal from being accomplished without first changing the mode. It is interesting and perhaps significant that these errors occur at a place in which the device makes a distinction (endMode versus startMode) that is not reinforced by prior knowledge.

Display-induced errors add a small physical cost and no benefit to programming the VCR. As might be expected by the principle of least-effort in operating the device, these errors decrease as participants gain experience with the VCR. Four of the seven display-induced errors occurred on the first performance show, one on the second, and two on the third. Hence, the least-cost method emerges gradually through routine interactions with the device. In this instance, the conflict between display-based difference-reduction and the task-to-device rule hierarchy is resolved in favor of the hierarchy.

Wrong Settings—Misencoding or Misretrieval Errors. Eleven psh-violates seem due to momentary lapses in task knowledge. These were cases in which participants set the display to the wrong channel or wrong 10min and either paused or did something else before detecting the error and setting the correct channel or 10min. For example, several errors involved participants setting the 10min display from “4” to “3” (toggling through 5, 0, 1, 2, and 3) and either pausing for longer than 2 s or doing something else before detecting the error and correcting the setting to “0.” These errors occurred despite the show information being visible.

One explanation for these errors is that participants misencoded the task knowledge. In the example above, perhaps the 10min information for start time was encoded as “3” rather than “0.” However, this assumption requires additional assumptions of either a later, correct, encoding of the 10min start time (from looking at the card again) and/or a later comparison of the time on the display to the time written on the 3×5 card. Whereas neither of these assumptions is unreasonable (and both are undoubtedly true in some circumstances), both require postulating special detection and correction mechanisms over and beyond those required for error-free performance. There is another possibility.

Three of these 11 cases involved setting the channel for the current show to the channel of a previous show. The other eight cases involved setting the start 10min to the end 10min of the current show. In three of these eight, the erroneous setting was also the start 10min of the previous show. In the other five cases, it was the end 10min of the previous show.

The presence in memory of other chunks encoding the channel or the 10min raises the possibility that the wrong chunk was retrieved. The full ACT-R theory maintains that memory is noisy in that the activation of a declarative memory element varies from moment to moment around its true value. If this were the case, then there would be occasions in which, for example, the most active 10min chunk would not be the correct 10min chunk. However, the next time the model glanced at the display, the random noise levels might be such that the chunk encoding the needed knowledge would have the higher activation and the error would be detected.

Three of the nine participants made no wrong settings. Of the six participants who

temporarily programmed the wrong channel or wrong 10min, four did so on one show only; two did so on two shows; no one made the error on all three shows. (Across shows, five errors were made on the first performance show, five on the second, and one on the third.) This pattern suggests that the error can be brought under strategic control. Indeed, the least-effort principle provides the motivation for doing so.

Each of the two hypotheses, misretrieval and misencoding, suggests a strategy by which the wrong setting error could be reduced. By the misretrieval hypothesis, the probability of misretrieving the current item can be reduced by strengthening the item at the time at which show information is originally encoded. The outlines of a theory of the use of such a strategy in serial attention is provided by Altmann and Gray (1999). By the misencoding hypothesis, strengthening a misencoded memory element would perpetuate not eliminate this class of errors. Rather, some sort of double-checking strategy would need to be employed to ensure that the encoded item is, indeed, the correct item.

Nothing in the current study can rule out either the misencoding or the misretrieval hypothesis. However, for the VCR paradigm, misretrieval provides an explanation of the origin of this class of errors that fits comfortably within the business-as-usual model of error detection and correction. In this sense, the misretrieval hypothesis permits the model to be simpler and more parsimonious than a model that requires constant visual recodings of the information on the 3×5 card and comparisons of that information to already encoded memory elements.

If either hypothesis is to become viable, it will be important to collect more such errors under circumstances in which one hypothesis or the other can be ruled out. It will also be important to expand the simple model used here into a more complex model of cognition. The simple model created for model-tracing supports neither misretrieval nor misencoding.

Miscellaneous Errors. Some of the remaining 15 psh-violates may fit in the above three categories. However, none of these occurred more than twice and each was idiosyncratic to a particular user. Eight errors (involving three participants) suggested transient differences between the participants' and the model's rule hierarchy. These included three cases in which each of three participants pressed the PROG key (to the left of the DAY OF WEEK key in Figure 2); two cases in which two participants pressed the PROG REC key at the wrong time; and three cases in which the CHANNEL key was pressed while in CLOCK SET mode. Seven of these eight transient rule hierarchy violations occurred during the first performance show. These violations incurred more effort for no benefit and, hence by the principle of cognitive least-effort, dropped out as the participant acquired a more efficient rule hierarchy.

Six errors were exhibited by one participant on her first or third performance show. Two of the errors involved the use of the CLEAR key (to the right of the MIN key in Figure 2) to reset the VCR after making an HOUR error. Used in startMode, the CLEAR key resets the display to CLOCK TIME. In both cases, the participant had toggled one HOUR too many (a pop-violate-postponed—discussed below). Rather than toggling the HOUR key 23 times, it was simpler to reset the display and start again. As the model does

not know about the CLEAR key, by the strict criterion used in this study, use of the CLEAR key is a violation of the control structure and hence an error. Another two errors made by this participant could be interpreted as double-checking the startMode settings and the endMode settings after they had been completed. However, two additional errors involved toggling between modes in a way that could not be easily interpreted.

Eight of the 15⁸ miscellaneous errors seemed due to transient rule hierarchy violations made by participants who were still learning the least-effort rule hierarchy. Four of the miscellaneous errors suggested knowledge or procedures that the model did not possess. However, by the strict criterion used in this study, all were violations of the control structure.

Summary of Push Errors. Of the 52 psh-violates (see Table 4), 19 were not true errors, but indicate that the participants' task-to-device rule hierarchy differed slightly from the model's task-to-device rule hierarchy. The transient nature of 8 of the 15 miscellaneous errors suggested that learning the least-effort control structure continued into the performance trial phase of the study. Four miscellaneous errors suggested knowledge or strategies not possessed by the model (and not used by other participants). The remaining push violates were examples of rule-based errors that presumably occurred because the wrong rule was evoked. The analytic framework provided by the model supported a display-based hypothesis regarding the nature of seven of these errors. Intriguing regularities led to the speculation that 11 of the errors were due to either misretrieval or to misencoding. The failure of the model to support such speculation argues that a more complete account of the origin of errors must await the development of a more complex model.

Pop Errors

Table 4 and Figure 7 indicate that 1041 of the 1059 pops matched what the model would have done to trace correct performance. Of these pop-matches, 984 were in service of one of the minimum number of goals required to program the VCR, 9 were in the service of error recovery goals, and 48 were classified as pop-misc. Pop-misc's are in service of various push errors. Pop-misc indicates that they are beyond the minimum set of pops that the model would need to successfully program the VCR. In addition, there are 18 pops that violate the control structure by popping prematurely and 30 instances that violate the control structure by not popping when they should. These 18 + 30 (48) pop-violates are the focus of the current section.

The Local Nature of Premature Pops. Pop-violate-prematures result in more work later by interrupting a correct goal before it is completed; that is, a correct goal is pushed but is then prematurely popped. For example, assume the current goal is SET-DOW, the target day is Saturday, and the current day is Tuesday. If the participant pressed the DAY OF WEEK button three times setting the VCR to Friday and then, before pressing DAY OF WEEK again, goes off and sets the channel, the SET-DOW goal has been prematurely popped. Before the VCR can be correctly set, the user has to repush the SET-DOW goal

(which is now classified as a psh-match-erRec) and press the DAY OF WEEK button one more time.

Tracing the above behavior requires the interruption of a goal before it is completed; that is, a premature pop. The experimenter stops the model immediately after the keypress that sets day-of-week to Friday and forces the current goal, SET-DOW, to pop. Going up one goal from SET-DOW (see Figure 6) puts the DO-startMode goal in control. At this point, the conflict set contains SET-channel, SET-DOW, and (assuming the start time has not yet been set) SET-startTime. As these are equal alternatives, the model would choose randomly among them; however, by following standard model-tracing procedure the experimenter resets the model to force it down the path chosen by the participant; namely, SET-channel. Hence, this premature pop requires traversing two links to a sibling goal. The pop-violate-premature classification turns what might be regarded as a sin of omission (e.g., neglecting one press of the DAY OF WEEK button) into one of commission (violating the task-to-device rule hierarchy); this violation of the task-to-device rule hierarchy needs to be explained.

The push/pop taxonomy in conjunction with the task-to-device rule hierarchy was used to determine the *nature* of the relationship between the interrupted goal and the interrupter. For two of the 18 interrupts, the interrupter was outside of the task-to-device rule hierarchy; these could not be considered.⁹ In three cases, two goals, a goal and its subgoal, were interrupted. For example, while programming the start hour, s02 interrupted himself to set the day-of-week. This interruption caused a premature-pop to occur for SET-start-hour and DO-startTime (see Figure 6). For the analysis of each of these three cases, the parent goal was omitted and only the distance between the old and the new terminal goal (usually these are both SET-*<goals>*) was tallied. In the example above, there are three links between SET-start-hour and SET-DOW (hence, the distance metric included the distance to DO-startTime, but DO-startTime was not counted as a separate premature pop). Such exclusions left a data set of 13 premature pops.

The distance between the interrupted goal and the interrupter as well as their relationship was calculated for each of the reduced set of premature pops. If it was a single goal and not the task of videotaping that was interrupted, an implicit prediction of the goal hierarchy is that the interrupter should have been close to the interruptee.¹⁰ This prediction appears to hold. For six of the 13 interrupts, the interrupter was a *sibling* goal (two-links away, sharing the same *parent* goal). For five of the interrupts the interrupter was an uncle or a nephew (three-links away). Finally for two of the interrupts the interrupter was a cousin (four-links away).

These data were compared to what would have been expected if the participants were randomly wandering around the task-to-device rule hierarchy. For each interrupted goal, a random-walk distance was calculated. This distance was the mean weighted distance that would have been expected if the interrupting activity were selected purely by chance. For example, if a participant interrupted herself while working on SET-DOW, what would be the mean weighted distance if an interrupter were chosen by chance? Figure 6 shows one goal (SET-channel) that is two-links away¹¹; three that are three-links away (SET-start-hour, SET-start-10min, and SET-start-min for DO-startTime); one that is four-links away

(SET-endMode); and five that are five-links away (SET-end-hour, SET-end-10min, and SET-end-min for DO-endTime; SET-csMode, and SET-PROGREC). The weighted mean distance of subgoals from SET-DOW is 4.00.

For the six cases where the interrupter was a sibling (two-links away) the mean weighted distance was 4.38 links. For the five cases where the interrupter was a nephew or uncle (three-links away), the mean weighted distance was 4.04 links. Finally, for the two cases where the interrupter was a cousin (four-links away), the mean weighted distance was 3.91 links. A paired-sign test on these 13 errors shows that the actual distance was consistently less than the mean weighted distance ($p = .006$).

The mean weighted distance used above includes goals for which the current state was visible as well as those for which it was not. For example, when the current goal was SET-start-hour, the state of the distant goal SET-end-hour was not visible. Perhaps rather than favoring goals that were close to the interruptee, participants were favoring goals whose current state was visible. This counter hypothesis suggested that if the calculation of mean weighted distance included only those goals whose states were visible at the time of the interrupt, there would have been little difference between choosing a visible goal at random (i.e., the mean weighted distance) and that calculated using the task-to-device rule hierarchy. This hypothesis was not supported by the data. For the six cases where the model classifies the interrupter as two-links away from the interruptee, the mean weighted distance of visible goals was 3.60. For the five cases classified as three-links away, the visible mean weighted distance was 3.54. Finally, for the two cases where the interrupter was four-links away, the visible calculation of mean weighted distance was 3.48. For 10 of the 13 errors the actual distance was less than the mean weighted distance whereas for three errors it was greater. Hence, when only the visible goals are considered, the actual distance was consistently and significantly (marginally) less than the random distance ($p = .09$).

Both metrics suggest that for the majority of cases participants were not mentally wandering around working on things at random. When one goal was interrupted, the goal that was taken up was usually close to it in the task-to-device rule hierarchy. This regularity provides further support for the task-to-device rule hierarchy adopted by the current model. For eight of the 13 cases (each of the six cases where the interrupter was two-links away and two of the five cases where the interrupter was three-links away) if the model was forced to prematurely pop the goal, the interrupter goal was a *legal* next move; one of a small number of goals that the model would consider and choose at random.

The Origin of Premature Pops. Unlike the push errors discussed earlier, the data suggest that premature pops increase as interactive behavior becomes more routine. Of the 13 premature pops, one was made on the first performance show, seven on the second show, and five on the third show. The potential importance of this rather ominous trend justifies speculation regarding its origins.

For push errors involving the wrong setting (discussed earlier), it seemed likely that information was either misencoded or misretrieved. In contrast, for premature pops, the participants did not stop at a likely misencoding or at one that had been used in a prior

show. For example, for one pop-violate-premature the participant stopped at start-10min “4” rather than “0,” for another the participant stopped when start-hr was at 18 rather than 23. For this small set of 13 premature pops, the only generalization that can be made is tautological: the VCR was closer to the goal state when the goal was prematurely popped than when it was pushed. Two alternative hypotheses go beyond this simple description of the data to suggest that the closer the VCR setting is to the goal setting the higher the probability of a premature pop.

ACT-R’s partial matching mechanism (Anderson & Lebière, 1998, p. 76–80) allows for the possibility that a chunk similar to the target chunk will be retrieved in place of the target chunk. This spurious retrieval permits a *goal completed* rule to fire, thereby popping the current goal before its true exit conditions have been met. Activation for a non-target chunk (one that does not completely match the retrieval specifications) will not be as high as for a target chunk but will increase as the similarity of the non-target to the target chunk increases. As memory is noisy, a chunk’s actual activation varies. Therefore, partial matching provides a mechanism whereby a less than perfect match can have a temporarily higher activation in the current context than a perfect match. A further assumption is that items (numbers and days of the week) that are closer to each other in serial position are more similar to each other (a notion for which there is some empirical support, see Lebière & Anderson, 1998). This implies that each step towards completing the goal will have a greater similarity to the goal state than did the preceding step. When this assumption is combined with partial matching it predicts that the probability of premature pops will increase the closer the participant gets to the goal state.

An alternative hypothesis requires going outside the strict goal-stack mechanism integral to ACT-R (e.g., see Altmann & Trafton, 1999). If goals are assumed to be simply another chunk in declarative memory, then it is possible that goals compete with each other. Indeed, display-induced push-errors (discussed above) were interpreted as illustrating competition among goals. Recent work in a serial attention paradigm (Altmann & Gray, 1998, 1999, 2000) predicts conditions under which the instruction for the current task loses activation as the task is performed. If the current goal behaves like any other element of declarative memory, then it may well be the case that it too loses activation as progress towards its completion progresses. Goal competition may be particularly problematic for artifacts in which the control structure is display-based and in which reminders of alternative goals are readily visible.

Both hypotheses allow for the possibility of competition from previous shows. The greater the number of prior shows, the greater the number of declarative memory elements that are associated with the goal of, for example, SET-DOW. The increased number of associations would produce a fan effect (see, e.g., Anderson & Lebière, 1998, p. 82–87) whereby the amount of activation available to the declarative memory element that encodes the current day-of-week is reduced. This reduced activation increases the probability of an error due to partial matching or to goal competition. Hence, neither hypothesis predicts that errors increase because interactive behavior becomes more

routine; rather, both predict that errors increase because memories from immediately prior shows persist.

The current data suggests that premature pops increased over shows. However, to establish this trend a much larger data set is required. The partial-matching and goal-competition hypotheses are intriguing; however, neither hypothesis can be readily incorporated into the simple model used here to trace correct behavior and to classify erroneous behavior. As was the case for push errors, accounting for the origin of premature pops must await the development of a more complex model.

The Nature and Origin of Postponed Pops. The 30 pop-violate-postponed errors include physical slips as well as cases in which the goal failed to be popped after it was achieved.

Twelve slips seemed readily attributable to simple physical problems manipulating the interface. Each of these 12 slips involved setting the mode (see left of Figure 2). Except for the mode switch, all other button objects on the simulated VCR required a simple click (a mouse down event followed by a mouse up event). Manipulating the mode switch required participants to mouse down and drag a button icon. This motion seems especially likely to be error prone and an occasion for true, physical slips.

Three errors involved double-clicking the same button immediately after entering the startMode (one error) or endMode (two errors). For the VCR, the first time the startMode or endMode is entered, all displays show EE:EE rather than time. In the startMode, the first click on any button initiates the mode by setting the displays to the current clock time, current day-of-week, and current channel. In the endMode, the first click on any button initiates the displays to the time and channel set in the startMode. The three errors all involved two clicks on the same time button (10min or hr). In each case, the time on the display, after the mode was initiated, was correct for that setting. For example, for a show that ran from 19:00 to 19:30, if the start time was correctly set, then once the end time was initiated the hr would be correctly set to 19. These three errors suggest that the participants were not using local display-based difference-reduction for these keypresses. Rather, the participants might have adopted a strategy of *keying ahead*. This strategy would reflect a bet on the part of the participants that the display would be off by at least one keypress and, therefore, that it would be reasonable to key ahead while waiting for perception to deliver the display's current setting to cognition.

Ten of the remaining 15 pop postponed errors involved setting the 10min setting to "0," none of the pop postponed errors involved setting the 10min to "3" (for 10min, "0" and "3" were used equally often, see Table 1). This error pattern suggests a conflict between the device-influenced rule hierarchy and the knowledge that participants brought to the device. Simply put, participants were uncomfortable counting 1 – 2 – 3 – 4 – 5 – 0. The cultural expectations are that "6" will follow "5" and that "0" will be preceded by "9." The scarcity of pop postponed errors for HOUR, MIN, CHANNEL, DAY OF WEEK, or 10MIN when the target setting was "3," suggests that participants adopted the strategy of keying rapidly when they were "far away" from the target setting but slowed down to monitor the results of every keypress when they neared the target. In cases where the

target was to set 10min to “0,” participants at “4” and “5” perceived that they were only half way to their target of “0.” As with the premature pops, these postponed pops did not diminish with practice. Four were made on the first performance show, none on the second, and six on the third.

Summary of Push and Pop Errors

By the criterion adopted in this paper, all violations of the model’s task-to-device rule hierarchy are errors. As the discussion of push errors showed, certain categories of errors suggested that individual participants had adopted a slightly different rule hierarchy than the model or were still in the process of acquiring the model’s least-effort rule hierarchy.

Seven push-errors were hypothesized to result from a conflict between display-based difference-reduction and the task-to-device rule hierarchy. Under certain circumstances, seeing an erroneous setting on the display or not seeing an expected setting sufficed to lure participants into making erroneous keypresses. This category of errors seemed to diminish as the task-to-device rule hierarchy became better practiced.

Eleven push errors were thought to result from temporarily using the wrong information. Whether the source of error was misencoding or misretrieval, the fact that this error decreased across shows suggests that people can adopt strategies that guard against it. The strategies for guarding against misretrieval are very different from those for guarding against misencoding. Determining which set of strategies were used would provide evidence supporting one of the two hypotheses.

Premature pops involved temporarily abandoning one goal to work on another. When these occur, the goal that was adopted tended to be close to the abandoned goal in the task-to-device rule hierarchy. Two untested hypotheses were put forward to explain premature pops. Both hypotheses predict that premature pops should increase as the distance to the goal state is reduced. Finally, in contrast to push errors, the evidence suggested that the rate of premature pops stayed constant or increased across shows.

Many postponed pops seemed explicable as simple physical slips involving one particularly problematic key. Of those that remained, the largest category suggested limits to participants’ reliance on local display-based difference-reduction. When the distance between the current setting and the goal setting was great, participants were hypothesized to key faster than the perceptual-cognitive-motor loop of local display-based difference-reduction would allow. Although this strategy may have worked successfully for most settings, its shortcomings were revealed in setting the 10min display to “0.” In this error filled case, the conflict between prior knowledge and device design was such as to cause participants to overshoot their goal before cognition became aware that it had been achieved. Similar to premature pops, postponed pops did not decrease with practice.

Although the data set of errors is small, the interpretations suggested by the pattern of errors are intriguing. Especially intriguing are the suggestions that one fundamental category of errors—push errors—may decrease with greater expertise; whereas the other fundamental category—pop errors—many increase. If further research involving larger

data sets of errors supports this conclusion, then work would need to be directed to engineering this pernicious category of errors out of the design of interactive systems.

Detection and Correction as Display-based Difference-reduction

Throughout the programming of the VCR, 28 goal or subgoal errors were made that if not detected and corrected would have prevented 20 of the 56 shows (36%) from being successfully programmed.¹² For these shows, all errors made were detected by the participants without the intervention of the experimenter. Can the model's place-keeping mechanism, display-based difference-reduction, detect these errors and are the rules contained in the model's task-to-device rule hierarchy sufficient to correct them?

Error detection for the model is business-as-usual. Global display-based difference-reduction works with the task-to-device rule hierarchy as follows. When the model is at DO-startMode, it scans the display to compare the day-of-week, channel, and start time of the display to the day-of-week, channel, and start time encoded in memory. If more than one discrepancy is noticed, it chooses at random. If the goal chosen is SET-DOW or SET-channel then local display-based difference-reduction works to keep pressing the key until the current setting matches the goal setting. If the chosen goal is DO-startTime, the model compares start hour, start 10min, and start min with the corresponding times encoded in memory. Again, if more than one discrepancy is noticed, it chooses at random, and relies on local display-based difference-reduction to keep pressing the key until the current setting matches the goal setting. The procedure is similar, though simpler, for DO-endMode.

Whenever the model is at DO-startMode or DO-endMode, global display-based difference-reduction checks all relevant settings. This checking is carried out even if an element of the display has been previously set. For example, having just pushed, correctly set, and popped SET-DOW the model is at DO-startMode. At this point, global display-based difference-reduction again directs the scan of the display to compare the current settings of channel, start time, as well as the just set day-of-week, to the values stored in memory.

The model will only detect errors when it is at DO-startMode, DO-startTime, DO-endMode, or DO-endTime. If an error is detected, it will correct it by doing exactly what it would do with any discrepancy. How well does this business-as-usual model account for the detection and correction of the 28 erroneous settings?

Seventeen of the detection episodes occurred at the parent node, one-link away from the discrepant node. The current model would have detected and corrected each of these.

Six of the detection episodes occurred at a grandparent node. For five of these the participant was at DO-startMode and detected a discrepancy between the displayed time and the target time (one-link away). Correcting this error entailed pushing the DO-startTime goal. At that goal, global display-based difference-reduction would have localized the discrepancy to SET-start-hour, SET-start-10min, or SET-start-min (one-link away). The sixth error involved the detection at the PROGRAM node of a discrepancy between the channel display and the target channel. Although this discrepancy was visible,

the model's global display-based difference-reduction would not have detected it at this point. Hence, neither the detection nor correction of this error can be explained by the current model.

Two of the detection episodes occurred at sibling nodes. These represent interrupts of one goal, for example, SET-start-hour, to correct a sibling goal, for example, SET-start-min. The current model cannot explain the interrupt; however, once the goal was interrupted, the model assumes the participant was at the parent node (e.g., DO-startTime). At that point, detection and correction of the discrepancy was routine.

Finally, in three cases the participant detected the discrepancy from somewhere outside the current model's task-to-device rule hierarchy. The model cannot account for these.

Of the 28 detection and correction episodes, the unmodified model gives a complete explanation of 22 (the 17 from the parent goal and five of those at the grandparent goal). This number increases to 24 if we include the two cases in which participants interrupt a goal to correct its sibling. The task-to-device rule hierarchy and display-based difference-reduction mechanisms postulated to explain the control of correct behavior can be applied, unmodified, to explain the detection and correction of error.

V. GENERAL DISCUSSION

Modeling the control structure by which people operate a simple, rule-based device required analyzing the constraints and opportunities presented by the embodied cognition, task, and artifact triad. This analysis was guided by three principles of cognitive engineering. The resultant model combined a task-to-device rule hierarchy with display-based difference-reduction.

Summary of the Nature, Detection, and Correction of Errors

The goal push/pop taxonomy (see Figure 7) categorizes the results of model-tracing the participants' action protocols. The fundamental distinction made by the taxonomy is between push errors and pop errors. For push errors, the participant attempted to do something that the model would not do. For pop errors, the participant failed to stop working on a goal when the model would stop.

Some of the push errors appeared to be display-induced, others were hypothesized to result from misretrieving or misencoding show information. For premature pops, the participant stopped before the setting matched the target setting. For postponed pops, the participant continued pressing the key after the target setting had been achieved.

Within these categories of error, important generalizations can be made. First, the detection of errors was a business-as-usual affair enabled by the normal control structure of the VCR. Global display-based difference-reduction worked within the confines of the task-to-device rule hierarchy. Most errors were detected at a place in the rule hierarchy where the model would have noticed a discrepancy between the setting of the display and its memory of the target setting. For most errors, the place at which a discrepancy was initially detected was a parent goal (one link away) of the error.

The local nature of error detection is congruent with Gray and Anderson's (1987) study of change episodes in programming. In studying the changes that programmers made to Lisp code after writing but before running the code, they found that changes either were made at the parent node—that is, before beginning a sibling goal—or rarely at all. For example, typos were either caught immediately after the word was typed or rarely at all. An embedded Lisp statement (i.e., an S-expression) would be changed immediately after it had been completed or rarely at all.

In both studies, the participant can reasonably be classified as being at the parent node when the change is initiated. In Gray and Anderson, changes were most likely to be limited to the just completed subgoal, whereas in the current study changes were likely to be made to any child node for which a discrepancy existed. Likely explanations for this difference between the two studies lie in the constrained order of Lisp code and the reliance on global display-based difference-reduction in programming the VCR. However, the similarities suggest error detection, whether in problem solving or routine performance, is a local affair.

Second, the correction of errors in this study was mostly business-as-usual. Special methods were not required. This business-as-usual correction seems more likely to occur for routine tasks than for problem-solving ones. (For example, for Lisp code, it seems unlikely that activities involved in debugging code will ever be identical to those that are involved in writing it.) However, even for routine tasks, business-as-usual correction of errors is not inevitable but must be designed into the artifact. Enabling the business-as-usual correction of errors seems to be a worthy design goal.

Third, when participants interrupted a not-yet-completed goal, they most often began work on one of its siblings. Most premature pops affected one level in the task-to-device rule hierarchy, some affected two levels; premature pops that traversed more than two levels of the task-to-device rule hierarchy were rare.

Fourth, conflicts between prior knowledge and the design of the device continued, and may have increased, throughout performance. Prior well-learned knowledge of counting may have induced postponed pops in setting the 10min display to "0." Counting 1 – 2 – 3 – 4 – 5 – 0 conflicted not only with prior knowledge but with device knowledge required to count the minutes and the hours.

Fifth, the pattern of pop postponed errors suggested that local display-based difference-reduction was one of two keypressing strategies. The error data suggested that when the current setting was far from the target setting (or was believed to be far from the target setting) that participants keyed faster than perception could deliver the results to cognition. This strategy resulted in errors when (as discussed above) prior knowledge conflicted with device design.

Sixth, conflicts between global display-based difference-reduction and the task-to-device rule hierarchy may have lead to display-induced push errors. These errors seem to have diminished as the task-to-device rule hierarchy became better learned.

Seventh, although the data are sketchy and far from conclusive, they suggest that push errors declined as interactive behavior became more routine whereas pop errors stayed constant or increased. Confirming this finding would have important implications for error

prevention. It would suggest that one fundamental category of errors can be reduced by training and experience whereas another cannot. Reducing pop errors would entail the discovery of additional principles of cognitive engineering and the development of methods to incorporate such principles into the design of interactive objects (see Gray & Boehm–Davis, in press).

Finally, the attempt to discover the nature of errors revealed the limits of the current model. The model's memory is perfect. Once encoded an item will not be forgotten. Neither misretrieval nor misencoding is possible. Once the model begins to do something, it will continue until its goal is achieved. Neither partial matching nor loss of activation can prematurely stop the model from accomplishing its goal. The model's strategies are limited as well. It only knows local display-based difference-reduction. It will never press keys faster than it can perceive and reason about the display. Similarly, when the model is at DO-endMode it will be totally focused on setting the end time. It will never be distracted by other items on the display. Remediating these shortcomings in the model is as necessary as collecting a more extensive set of data to the further understanding of the nature, detection, and correction of errors.

Principles of Cognitive Engineering

The interactive behavior required to program the VCR emerged through a combination of constraints and opportunities provided by the interaction of embodied cognition with the task and artifact designed to accomplish the task (see Figure 5). These constraints and opportunities were manifested in three principles of cognitive engineering; least-effort in operating the device, least-effort in mapping prior knowledge to device knowledge, and least-effort in place-keeping.

The design of the VCR determined the cost of alternative combinations of cognitive, perceptual, and motor operators that could have been used to program the VCR. Although many of these alternatives would have resulted in successfully programmed shows, participants generally conformed their interactive behavior to the least-cost sequence of motor movements defined by the design of the artifact. The design reduced the cognitive cost of place-keeping by supporting display-based difference-reduction.

During the instructionless learning phase of the study, prior knowledge of electronic devices and knowledge of English worked with the displays to enable a label-following strategy (Polson & Lewis, 1990). In the absence of overt instruction, participants were able to eliminate most nonfunctional keys from their keypressing repertoire.

In terms of the task-to-device rule hierarchy, prior knowledge caused participants to treat three adjacent keys (HOUR, 10MIN, and MIN) as three subgoals of a DO-startTime or DO-endTime goal.

In contrast, least-effort in operating the device conflicted with prior knowledge in several instances. Prior knowledge of the task (as per Figure 1) conflicted with the moded structure imposed by the device. During instructionless learning, most participants did not correctly use the modality switch or the PROG REC button. During the performance phase, errors involving modality or PROG REC were a major source of errors outside the

model. Another conflict was in the operation of the 24-hr clock. Many participants had trouble translating AM and PM times into their 24-hr equivalents. For example, some had some trouble deciding if half past midnight should be represented as 00:30 or 12:30.

During performance, both local and global display-based difference-reduction were supported as least-cost strategies for place-keeping. Once a SET-<goal> was begun, each keypress brought the user one key closer to the target state. Users did not have to calculate the difference between the current state and the target state, they merely had to keep pressing the key until the target state was achieved.

The above description suggests that local display-based difference-reduction is governed by a perceptual-cognitive-motor loop; see the display, decide that the current state is not the target state, press the key. However, the error data suggested that the truth is more complex and that a two-tier strategy was adopted. When participants thought that the current state was far from the target state, they seemed to have keyed faster than the perceptual-cognitive-motor loop would allow. In contrast, when the current state was believed to be close to the target state then local display-based difference-reduction was employed.

The design of the VCR partially implemented global display-based difference-reduction. In START mode, the start time, channel, and day-of-week settings were visible. In END mode, the end time and channel settings were visible. Most of the cognitive burden for remembering which goals had been achieved and which were yet to be accomplished was removed from cognition and placed onto a simple perceptual-cognitive loop. This loop sufficed to make error detection a business-as-usual process for this device.

Design Issues

This paper presented a detailed analysis of the interactive behavior for a particular task and artifact. What lessons does this analysis yield for the design of the next artifact for the same or different routine task?

The most general lesson is that the end result of the design process is not an artifact but a set of interactive behaviors that are shaped by embodied cognition interacting with a task and an artifact designed to accomplish the task (see Figure 5). Unfortunately, whereas this observation may sound like a truism to cognitive scientists weaned on Simon's ant-on-the-beach metaphor (Simon, 1996), to designers it is likely to seem as vacuous as the slogan "know thy user." Treating the above as a case study, what more specific lessons-learned can be derived?

First, display-based difference-reduction is a powerful tool for place-keeping. However, it needs to be introduced to designers as a design goal. A casual survey indicates that often, as in the current study, it is partially implemented. For the commercial VCR, this partial implementation had two negative consequences. In end mode, it led to display-induced errors. In both modes, it required that some of the burden for place-keeping regarding the other mode be imposed on cognition. For example, in end mode, participants had to remember that the start time and day-of-week had (or had not) been programmed. In start mode, they had to remember that end time had not been

programmed. Although this extra cognitive burden was not postulated to be a major factor in the errors found in the current study, under conditions of higher cognitive workload, it might well be.

Second, device-specific goals are the hardest for participants to remember to accomplish. In this study, many of the trials that were not successful failed because of the post-completion error of forgetting to set the VCR to PROG REC. If such device-specific goals are required, the device should be designed in such a way as to make their accomplishment unavoidable. An extreme solution to this problem is represented by the modal dialog boxes¹³ that are thrust on the user by many GUI interfaces.

Third, task-goals should be mapped onto the artifact in a way that is consistent with the user's pre-existing task knowledge. (The call for designers to attend to how prior knowledge is mapped onto the current artifact has been sounded before; for example Moran, 1983; Payne, Squibb, & Howes, 1990; Young & Whittington, 1990). For the current VCR the participant was required to acquire two new goals, DO-startMode and DO-endMode, that imposed a new structure on the task-goals of start time, end time, day-of-week, and channel. Failure to use the modes was a common problem during instructionless learning. During the performance phase, mode errors prevented many shows from being successfully programmed (see Table 5). Finally, for trials on which errors were made but detected and successfully corrected, display-induced push errors were diagnosed as resulting from a conflict between global display-based difference-reduction and the mode structure.

As another example, the commercial VCR fragmented the 24-hr clock into three subgoals—SET-<start/end>-hour, SET-<start/end>-10min, and SET-<start/end>-min. This fragmentation presented participants with an unfamiliar way of thinking about time. Likewise, the expectation that "0" would be preceded by "9" rather than "5" was postulated as responsible for the large number of pop-violate-postponed errors involving setting the 10min display to "0." Unfortunately, pre-existing knowledge may directly conflict with the goal of minimizing user effort or other design considerations. At present, such tradeoffs between learning and performance cannot be derived from theory; however, these tradeoffs can be cast as empirical questions that can be resolved by empirical testing.

Fourth, the results of the current study as well as from Gray and Anderson (1987) suggest that if an error is caught it will be caught locally. For all systems, this suggests that error detection should shortly follow error creation. However, for display-based systems, the current study suggests that this rule can be relaxed in that error detection does not need to occur after every goal, but before a goal's parent goal is popped. This distinction is somewhat subtle and clearly needs supporting research before it can be established as a design principle.

Fifth, although errors may seem rare when compared to the total number of correct actions, they are important. Whereas only 4% of the keypresses made while programming the VCR were mistakes or slips, if any one of these keypresses had not been detected and corrected, the VCR would not have been successfully programmed. Problems with just 4% of the keypresses would have prevented shows on 37 of the 56 trials (66%) from being

successfully recorded. A misprogrammed show is a minor annoyance to the user. However, devices with the approximate complexity of a VCR are ubiquitous and have found their way into emergency rooms, airplane cockpits, power plants, and so on. Errors of ignorance may be reduced by training; however, errors in the routine performance of skilled users can only be reduced by design.

VI. CONCLUSIONS

It requires a deep understanding of the interactions among embodied cognition, task, and artifact to analyze the interactive behavior required for the performance of even simple tasks. As the current effort has demonstrated, deliberately limiting two elements of the triad by using a simple task and a common artifact only highlights the complexity of the third element, embodied cognition.

A common tactic in cognitive science is to limit the degrees of freedom available to the modeler by showing how the process model can be acquired; that is, how a novice system can acquire the knowledge required for expert performance. A complementary tactic was adopted here. For the current study, a large data set of mostly correct behavior was collected and model-tracing was used to catalog each violation of behavior from the model. Rather than focus on the large amount of correct behavior that the model explained, the burden of proof for the model was cast as how well it could account for violations of itself.

The findings are that a simple task-to-device rule hierarchy when combined with display-based difference-reduction works surprising well. Not only can such a control structure account for correct performance, but it also yields an account of the nature, detection, and correction of errors (both overt as well as covert). The success of this effort supports the application of least-effort principles of cognitive engineering to the design of interactive devices.

Acknowledgments: This research was supported by grants from the Office of Naval Research (#N00014-95-1-0175) and the National Science Foundation (IRI-9618833). I thank Haresh Sabnani for his work in programming the original VCR device simulation. Special thanks are due to Erik M. Altmann for his careful reading and detailed comments on several versions of this paper. Thanks are also due to Deborah A. Boehm-Davis, Michael D. Byrne, Clayton Lewis, John R. Anderson, and Stephen J. Payne for their comments on earlier versions of this paper. A much abbreviated report on this work appeared in the Proceedings of Interact'95 (Gray, 1995).

NOTES

1. GOMS is an acronym for the elements of goal-based cognition: goals, operators, methods, and selection rules.
2. Information concerning the object clicked (VCR button), the time was clicked, and the current state of the VCR were saved to a log file.

3. Participants used a mouse to interact with the simulation. The actual VCR was operated by pressing and sliding various physical buttons. Hence, neither the simulated nor the actual VCR required keypresses. Few task analysis methods analyze behavior down to the level of physical actions (see, e.g., the survey of task analysis methods reported by Kirwan & Ainsworth, 1992). Throughout this paper, my use of the terms “keypress” and “keystroke” reflects the fact that by including mouse clicks (or button presses) in the analysis, the task analysis is at the “keystroke level.” This usage of the term “keystroke level” follows the distinction made by Card et al. (1983).
4. People may know many things about a show that might be relevant to the task of recording it. For example, users may know which shows precede and follow the target show. Likewise, they may know the duration of the target show or a show’s VCR-plus™ number. The first difficulty in using the VCR is to determine what subset of task knowledge the particular device requires.
5. Various mechanisms for this emergence can be postulated. The one favored in this proposal is that which is incorporated into ACT-R (Anderson & Lebière, 1998). For ACT-R, rules are selected on each cycle based on their expected value. Expected value is determined by subsymbolic mechanisms that weigh a rule’s probability of success against its cost of execution. (An extended discussion of this mechanism is beyond the scope of the current paper.)
6. For the model, using global display-based difference-reduction to localize a discrepancy with a time setting is a two-step process. First the discrepancy between the display time and target time is noted, then the discrepancy is localized to hour, 10min, or min.
7. Much of ACT-R, the theory, is devoted to explaining changes in activation of declarative memory elements. However, the models used here did not incorporate those features of ACT-R.
8. Fourteen of the 15 miscellaneous errors are discussed in the text. The fifteenth error involved revisiting and resetting the HOUR after it had been correctly set.
9. In both cases, the same participant, s07, pressed the CLEAR key which reset the start time to the clock time.
10. If the task of videotaping were interrupted by, for example, a phone call or a screaming child, then the business-as-usual model should be able to trace performance on task resumption. However, under those circumstances, the user would not be expected to pick up near where s/he left off.
11. Note that SET-startMode is also two links away but is NOT counted towards the mean weighted distance. Any change in the mode switch would be counted as either SET-endMode or SET-csMode.
12. This estimate excludes postponed pops. With these included, the percentage of unsuccessful shows would increase to 67%.
13. A modal dialog box is one of those annoying messages from your computer that will not go away and will not let you do anything else until you select one of the options it presents.

REFERENCES

- Allwood, C. M. (1984). Error detection processes in statistical problem solving. *Cognitive Science*, 8, 413–437.
- Allwood, C. M., & Bjorhag, C. G. (1990). Novices debugging when programming in Pascal. *International Journal of Man-Machine Studies*, 33(6), 707–724.
- Allwood, C. M., & Bjorhag, C. G. (1991). Training of Pascal novices error handling ability. *Acta Psychologica*, 78(1–3), 137–150.
- Altmann, E. M., & Gray, W. D. (1998). Pervasive episodic memory: Evidence from a control-of-attention paradigm. In M. A. Gernsbacher, & S. J. Derry (Eds.), *Twentieth annual conference of the Cognitive Science Society* (pp. 42–47). Hillsdale, NJ: Erlbaum.
- Altmann, E. M., & Gray, W. D. (1999). Serial attention as strategic memory. In *Twenty-First Annual Conference of the Cognitive Science Society* (pp. 25–30). Hillsdale, NJ: Erlbaum. .
- Altmann, E. M., & Gray, W. D. (2000). Preparing to forget: Memory and functional decay in serial attention. *Manuscript in preparation*.
- Altmann, E. M., & Trafton, J. G. (1999). Memory as goal store: An architectural perspective. In *Twenty-First Annual Conference of the Cognitive Science Society* (pp. 19–24). Hillsdale, NJ: Erlbaum.
- Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.
- Anderson, J. R. (1991). Is human cognition adaptive? *Behavioral and Brain Sciences*, 14, 471–517.
- Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.
- Anderson, J. R. (1995). *Cognitive psychology and its implications* (4th ed.). New York: W. H. Freeman.

- 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., & Jeffries, R. (1985). Novice LISP errors: Undetected losses of information from working memory. *Human-Computer Interaction*, 1(2), 107–131.
- Anderson, J. R., & Lebière, C. (Eds.). (1998). *Atomic components of thought*. Hillsdale, NJ: Erlbaum.
- Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science*, 2, 396–408.
- Ballard, D. H., Hayhoe, M. M., & Pelz, J. B. (1995). Memory representations in natural tasks. *Journal of Cognitive Neuroscience*, 7(1), 66–80.
- Booth, P. A. (1990). Identifying and interpreting design errors. *International Journal of Human-Computer Interaction*, 2(4), 307–332.
- Brown, J. S., & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 4(4), 379–426.
- Brown, P. S., & Gould, J. D. (1987). An experimental study of people creating spreadsheets. *ACM Transactions on Office Information Systems*, 5(3), 258–272.
- Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science*, 21(1), 31–61.
- Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.
- Cuniff, N., Taylor, R. P., & Black, J. B. (1989). Does programming language affect the type of conceptual bugs in beginner's programs? A comparison of FPL and Pascal. In E. Soloway, & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 419–429). Hillsdale, NJ: Erlbaum.
- Ericsson, K. A., & Lehmann, A. C. (1996). Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual Review of Psychology*, 47, 273–305.
- Ericsson, K. A., & Smith, J. (Eds.). (1991). *Toward a general theory of expertise: Prospects and limits*. New York: Cambridge.
- Fitts, P. M., & Posner, M. I. (1967). *Human performance*. Belmont, CA: Brooks Cole.
- 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., & Boehm-Davis, D. A. (in press). Milliseconds Matter: An introduction to microstrategies and to their use for interactive devices. *Journal of Experimental Psychology: Applied*.
- Huguenard, B. R., Lerch, F. J., Junker, B. W., Patz, R. J., & Kass, R. E. (1997). Working memory failure in phone-based interaction. *ACM Transactions on Computer-Human Interaction*, 4(2), 67–102.
- John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction*, 3(4), 287–319.
- Johnson, P. E., Duran, F., Hassenbrock, A., Moller, J., Prietula, M., Feltovich, P., & Swanson, D. (1981). Expertise and error in diagnostic reasoning. *Cognitive Science*, 5, 235–283.
- Kirwan, B., & Ainsworth, L. K. (Eds.). (1992). *A guide to task analysis*. Washington, DC: Taylor & Francis.
- Lansdale, M. W. (1995). *Modeling errors in the recall of spatial location* (Technical Report): Cognitive Ergonomics Research Group Loughborough University of Technology.
- 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: Erlbaum.
- Lebière, C., & Anderson, J. R. (1998). Cognitive arithmetic. In J. R. Anderson, & C. Lebière (Eds.), *Atomic components of thought* (pp. 297–342). Hillsdale, NJ: Erlbaum.
- Lerch, F. J., Mantei, M. M., & Olson, J. R. (1989). Skilled financial planning: The cost of translating ideas into action. In K. Bice, & C. Lewis (Eds.), *ACM CHI'89 conference on human factors in computing systems* (pp. 121–126). New York: ACM Press.
- Lohse, G. L., & Johnson, E. J. (1996). A comparison of two process tracing methods for choice tasks. *Organizational Behavior and Human Decision Processes*, 68(1), 28–43.
- Moran, T. P. (1983). Getting into a system: External-internal task mapping analysis. *Proceedings of CHI'83 Conference on Human Factors in Computing Systems* (pp. 45–49). New York: ACM.
- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

- Nooteboom, S. G. (1980). Speaking and unspeaking: Detection and correction of phonological and lexical errors in spontaneous speech. In V. A. Fromkin (Ed.), *Errors in linguistic performance: Slips of the tongue, ear, pen, and hand* (pp. 87–95). San Francisco: Academic Press.
- Norman, D. A. (1981). Categorization of action slips. *Psychological Review*, 88(1), 1–15.
- 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, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker*. New York: Cambridge University Press.
- Payne, S. J., Squibb, H. R., & Howes, A. (1990). The nature of device models: The yoked state space hypothesis and some experiments with text editors. *Human-Computer Interaction*, 5(4), 415–444.
- Polson, P. G., & Lewis, C. H. (1990). Theory-based design for easily learned interfaces. *Human-Computer Interaction*, 5(2–3), 191–220.
- Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3), 257–266.
- Reason, J. (1990). *Human error*. New York: Cambridge University Press.
- Rizzo, A., Bagnara, S., & Visciola, M. (1988). Human error detection processes. In E. Hollnagel, G. Mancini, & D. D. Woods (Eds.), *Cognitive engineering in complex dynamic worlds* (pp. 99–114). New York: Academic Press.
- Rogers, R. D., & Monsell, S. (1995). Costs of a predictable switch between simple cognitive tasks. *Journal of Experimental Psychology: General*, 124(2), 207–231.
- Seifert, C. M., & Hutchins, E. L. (1992). Error as opportunity: Learning in a cooperative task. *Human-Computer Interaction*, 7(4), 409–435.
- Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge, MA: The MIT Press.
- Smelcer, J. B. (1995). User errors in database query composition. *International Journal of Human-Computer Studies*, 42(4), 353–381.
- VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology*, 47, 513–539.
- VanLehn, K. A. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.
- Xiao, Y., Mackenzie, C. F., & Group, L. (1995). *Decision making in dynamic environments: Fixation errors and their causes*. Paper presented at the Human Factors and Ergonomics Society 39th Annual Meeting, Santa Monica, CA.
- Young, R. M., & Whittington, J. (1990). Using a knowledge analysis to predict conceptual errors in text-editor usage. In J. C. Chew, & J. Whiteside (Eds.), *ACM CHI'90 conference on human factors in computing systems* (pp. 91–97). New York: ACM Press.