GIT-COGSCI-95/XX

A Working Memory Model of a Common Procedural Error

Michael D. Byrne
Susan Bovair

July, 1995

Michael D. Byrne, Susan Bovair
School of Psychology
Georgia Institute of Technology
Atlanta, GA 30332-0170
byrne@cc.gatech.edu, susan.bovair@psych.gatech.edu

Acknowledgements: We would like to thank Tim Salthouse, David Kieras, and Scott Wood for their comments on an earlier draft. The first author would also like to thank the National Science Foundation for financial support while much of this works was conducted.

Abstract

Systematic errors in performance are an important aspect of human behavior that have not received adequate explanation. One such systematic error is termed post-completion error; a typical example is leaving one's card in the automatic teller after withdrawing cash. This type of error seems to occur when people have an extra step to perform in a procedure after the main goal has been satisfied. The fact that people frequently make this type of error, but do not make this error every time, may best be explained by considering the working memory load at the time the step is to be performed: the error is made when the load on working memory is high, but will not be made when the load is low. A model of performance in the task was constructed using Just and Carpenter's (1992) CAPS that predicted that high working memory load should be associated with post-completion errors. Two experiments confirmed that such errors can be produced in a laboratory as well as a naturalistic setting, and that the conditions under which the CAPS model makes the error are consistent with the conditions under which the errors occur in the laboratory.

Introduction

In their everyday interaction with the world, people often make mistakes, slips, lapses, miscalculations and the like. Although making errors is common and the effects of errors can range from the merely annoying to the catastrophic, procedural errors have received relatively little attention from cognitive psychologists. Senders and Moray (1991, p. 2) suggest that "[o]ne reason for this is that error is frequently considered only as a result or measure of some other variable, and not as a phenomenon in its own right." Typically, procedural errors are viewed as the result of some stochastic function. Decrements in performance are manifested as an increase in global error rate, with little or no attention paid to what causes any particular error.

This stochastic view of error does not seem to correspond to people's intuitions about their own behavior. People seem prone to making some kinds of errors more often than others, and errors seem to occur more often at particular steps in the execution of a given procedure. In some cases, errors are simply the result of systematic deficiencies or "bugs" in knowledge (e.g. Brown & VanLehn, 1980). That the lack of correct knowledge of how to perform a step would lead to errors on that step seems a plausible explanation of some kinds of error. However, this explanation does not cover cases where people do have the correct knowledge. Many people report making errors such as leaving the original document behind in a photocopier, failing to retrieve one's bank card from an automated teller machine (ATM), and forgetting to replace the gas cap after filling the tank. In all these cases, people almost certainly have the knowledge required to carry out the task, because they perform the task correctly most of the time. Yet errors in these and other similar tasks are often reported.

There appears to be a class of common errors where people have the knowledge required to perform the task correctly, but still fail on occasion. One kind of these errors seems to occur on tasks with a similar task structure. The key characteristic of this task structure is that some action (retrieving the original, replacing the gas cap) is required to be performed after the main goal of the task (get copies, fill up) has been satisfied or completed. Thus, this type of error will be referred to as post-completion error. This paper will attempt to answer two key questions concerning post-completion error: Are post-completion errors special in some way or are they just a result of a generally high error rate in particular tasks? What is it that causes post-completion errors? While the answer to the first question has not, to this point, been addressed, there has been some work done in trying to understand the source of post-completion errors.

Previous Approaches to Post-Completion Error

Two different explanations for post-completion error have been proposed by researchers in human-computer interaction (Young, Barnard, Simon, & Whittington, 1989; Polson, Lewis, Reiman, & Wharton, 1992); we will label these the goal stack approach and the goal killoff

approach. Because both of these accounts rely heavily on references to task structure, it will be useful to provide an example of the task structure of a typical task that is subject to post-completion error. One of the prototypical post-completion tasks is using a photocopier. The main goal of using a photocopier is to get one or more photocopies. Before actually getting the copies in hand, the photocopier user is typically required to execute one or more steps, such as placing the original document on the glass and indicating the number of copies to be made. The copier then produces the desired copies, and the main goal is satisfied. The post-completion error here is to walk off with the copies but forget the original document, which still sits on the glass under the cover. Figure 1 depicts one representation of the task structure for a typical photocopier task. Ovals represent goals, and rectangles represent operators. Arrows connecting goals represent goal-subgoal relations and arrows connecting operators represent action sequences. Arrows connecting goals and operators show the action sequences associated with particular goals. Subgoals for a goal are executed in a left-to-right order.

In determining the source of post-completion errors, it is important to distinguish between the aspects of task structure that are determined by the device and those which are a function of a person's goals. When using a simple device such as a photocopier, the user typically starts with only one task relevant goal: getting copies. However, if the user is to successfully operate the device in service of this goal, they will have to generate subgoals according to the task structure inherent in the device. Thus, the user will typically follow the procedural path required by the device. At the remove copies operator, however, the path required by the device and that which is necessary to satisfy the user's goal diverge. (The double vertical lines in Figure 1 serve to emphasize that operator.) When the copies are removed from the copier, the user has fulfilled the get copies goal—he or she now has copies, so the get copies goal is no longer relevant. From the user's point of view, the goal relevant to the photocopier is satisfied and the next goal may be pursued. However, from the perspective of the device, the task is not complete—the original document has not yet been removed. That is, the execution path for the user and the device diverge. The divergence is represented by the dashed arrow in Figure 1. What needs to be explained is the reason that this divergence might lead to error (as has been observed by Norman, 1980).

One account developed by Young and his colleagues (Young et al., 1989) we will call the goal stack approach because it employs the idea of a goal stack to explain the source of the error. Stacks are used to manage goals in a number of computational models, most notably ACT-R (Anderson, 1993) and Soar (Laird, Newell, & Rosenbloom, 1987). The way a stack works is quite simple: when a new goal is created, it is placed onto the top of the stack (or "pushed"), becoming the current goal. Subgoals generated by a goal on the stack are also pushed onto the stack. When a goal is satisfied, it is removed from the stack (or "popped"), making what was the

next goal down on the stack the current goal.

It is possible in the course of pursuing a subgoal that not only will that subgoal be satisfied, but one of the goals further down in the stack will be satisfied; thus, satisfying some subgoals may satisfy a parent goal, or supergoal, as well. In that case, a strict goal stack stipulates that the supergoal and everything above it in the stack are popped. This leaves the system free to pursue whatever was below that supergoal on the stack.

In the photocopier example, for instance, the supergoal might be get copies. This goal would be pushed onto the stack. Next, a place paper subgoal would be pushed, then popped when the original is successfully placed. A subgoal like make copies as well as the get copies supergoal will be on the stack when the copier spits out the copies, and both the subgoal of making copies and the supergoal of getting copies are satisfied. Thus, the make copies goal—and any associated subgoals—will be popped because it is satisfied. The user forgets to retrieve the original document because the remove original subgoal must be on the stack for this retrieval to occur, but at the appropriate time there are no goals relevant to the photocopier on the stack. In general, the goal stack approach claims that post-completion errors arise because subgoals are lost when the main goal for a task is popped before the task is actually complete.

Although this explanation has intuitive appeal, it does not make entirely plausible predictions. According to this account, people should make this error every time—the goal structure for tasks like the photocopier never changes, so the error should persist indefinitely. Of course, the goal stack account can be "patched" to eliminate the error by stipulating that the goal to perform the post-completion step gets placed on the stack. This would solve the problem in that a stack-managed system would perform the task correctly, but it would now perform the task correctly on every single trial, which is just as poor a prediction as making the error on every single trial.

One might hypothesize that a person begins with the non-patched version of the procedure and learns the patch. This account predicts that people should transition from always making this error to never making this error. If this account were correct, people should leave the original on the photocopier the first few times they use it, and once they get it right they never make the post-completion error again. These predictions have yet to receive empirical support, and do not seem to match most people's intuitions about making post-completion errors.

A related but slightly different account has been developed by Polson and colleagues (Polson et al., 1992) and is termed the goal killoff approach. This account is based on the construction-integration (C-I) model (Kintsch, 1988), and uses the term "supergoal kill-off" instead of post-completion error. The construction-integration model was originally developed to model text comprehension, but has been applied in other areas as well (e.g. Mannes & Kintsch,

1991; Kitajima & Polson, 1992). The construction-integration model is a hybrid symbolic/connectionist system, and its account of post-completion error depends on inhibition of goals. C-I manages goals by using inhibition from done-it nodes to goal nodes. Every goal in C-I has associated with it a done-it node. The conditions for satisfying a goal are represented in the done-it node, which becomes active when those conditions are met. When a done-it node becomes active, it inhibits the goal to which it is attached.

In this account, post-completion error arises because the done-it node for the main goal (get copies) is very similar to the done-it node of the make copies subgoal. The system then confuses the two done-it nodes and activates the done-it node for the main goal. When the done-it node for the main goal becomes active, it inhibits the main goal. When a goal is inhibited, all related subgoals are inhibited, so this causes the remove original subgoal to lose activation and it therefore never gets executed.

There are also some problems with this approach. The major theoretical problem is that similar is never operationalized; how similar do two goals have to be to generate the error? Though a variety of operationalizations of similarity have been presented by various researchers in other contexts, no such notion has been incorporated into C-I, nor is it clear how similarity metrics would apply to goals in C-I. In addition, it is not clear what the performance implications would be if C-I were constantly computing similarity between all active goals. While this does not rule out a similarity-based account of the phenomena, it makes the offered explanation weak. The second difficulty with the C-I account is that it has the same problem the goal stack account does: if the two goals are similar, then the error should always be made, if the goals are not similar, the error should never be made.

In both the goal stack and goal killoff accounts, the post-completion error occurs because the relevant subgoal is lost, suggesting that management of goals is responsible for the error. So, while looking at extant accounts of post-completion errors suggests that the explanation can be found in goal management, they do not offer a satisfactory mechanism for realistically generating post-completion errors, nor have they successfully stimulated empirical results. Because these two accounts are not satisfactory, a better account is necessary. One way of pursuing such an account is to derive it from a more general theory of error.

Previous Work on General Error

One of the most comprehensive well-known works in error research is James Reason's (1990) book Human Error. He divides errors into three classes, corresponding to the three levels of behavior he describes: skill, rule, and knowledge. A significant bulk of the theory he proposes deals with errors based on incomplete or incorrect knowledge, and therefore at the knowledge level. He does, however, give consideration to non-knowledge-based errors, those at the "skill"

level. These errors are termed slips or lapses, and are characterized by “failure to make an attentional check” or “failure to monitor the current intention” (p. 60, 61). Errors at higher levels, rule and knowledge, are termed mistakes, which occur “at the level of intention formation.”

Reason would probably classify post-completion errors as slips, which he defines as “errors which result from some failures in the execution and/or storage stage of of an action sequence” (p. 9). Unfortunately, Reason does not go into much detail about how slips emerge from the human cognitive system. In his framework, they come from either inattention or overattention; people either fail to provide an attentional check when they should or they perform one when they should not. Reason would presumably explain post-completion errors as resulting from inattention (omitting the attentional check after fulfilling the main goal), but this explanation begs the question of the source of the error: where do the attentional failures come from? Reason does not describe in much detail how the cognitive system would be systematically prone to these attentional glitches. They just seem to occur, possibly stochastically, possibly not—no further explanation is provided.

In his classic paper on errors, Norman (1981) defines a slip as “a form of human error defined to be the performance of an action that was not what was intended” (p. 1). That is, a slip is a mis-execution of a correct procedure—exactly the kind of error in question. This paper primarily provides a brief taxonomy of different kinds of slips, based on naturalistic observations and self-reports of errors. Norman does, however, begin the paper with some thoughts about how errors might emerge. He describes a model based on an activation-trigger-schema (ATS) system. In such a system, various schemata (organized memory units) for action exist at varying levels of activation. Each schema has a series of triggering conditions associated with it, much like the if side of a production rule. He assumes that, at any given moment, multiple schemata will be active and that schemata will be applied according to some function that trades off between activation level and goodness-of-match of the trigger conditions, but he does not describe this trade-off in more detail. In this account, errors typically emerge either because the activation of the correct schema was too low or because the goodness of match to some incorrect one was too high. While this may begin to suggest a mechanism for errors in correct knowledge situations, it does not precisely specify the conditions under which errors are likely. In fact, this explanation is closely related to the goal killoff explanation (Polson et al., 1992) in that it relies on an incompletely-specified definition of similarity. When is a match “good” in the ATS scheme? Because this is not specified, the A TS scheme does not help much in error prediction.

Baars (1992a) has also undertaken an extensive research program on errors. Though this program concentrated on speech errors, Baars (1992b) suggests that the same mechanisms

responsible for speech errors cause other action errors as well. In particular, he suggests that slips occur because "[c]ompetition between two alternative plans may also overload the limited-capacity system momentarily, thereby interfering with the conscious access that is needed to resolve nonroutine choices caused by the competing plans" (p. 30). His conceptualization, however, neither directly addresses post-completion errors nor contains enough specification to predict their occurrence or non-occurrence with much precision.

All of these approaches to a general theory of error are subject to criticism because of their lack of specificity; they simply do not provide enough explanatory power to make specific predictions. In contrast, the computational accounts are more specific but they have not led to successful empirical demonstrations. There are at least two potential reasons for this. Post-completion errors may not, in fact, be caused by any of the supposed mechanisms but may simply be stochastic in nature. Alternatively, the proposed mechanisms may not be the ones responsible for post-completion errors. The lack of laboratory data on this phenomenon makes it difficult to determine which is the better explanation. Experiment 1 is an attempt to generate post-completion errors in the laboratory so that possible explanations can be explored.

Experiment 1

The primary motivation behind Experiment 1 was to try to get participants to make post-completion errors in a laboratory setting. Producing systematic procedural errors in a laboratory setting is a rare event; as Sellen and Norman (1992, p. 334) point out, "the laboratory environment is possibly the least likely place where we are likely to see truly spontaneous, absentminded errors." For a variety of reasons, probably the most critical of which is self-monitoring on the part of the participants, laboratory studies in which such errors are produced are few and far between. Experiment 1 represents an attempt to develop an experimental paradigm in which procedural errors are frequent enough to be studied in detail. We attempted to discourage self-monitoring in two ways: (1) by framing the tasks in an engaging fictional setting so participants would focus less on the laboratory setting, and (2) by encouraging subjects to work quickly and compete with their own past performance.

A second aim of Experiment 1 was to see if there was anything to indicate that post-completion errors are different from other errors. If all procedural errors are caused by the same mechanism, errors should occur with equal frequency on all steps. On the other hand, if post-completion errors are reliably more or less frequent, this would imply that they are unusual in some way. One of the problems with assessing error frequency in many naturalistic tasks is the possible non-equivalence of the different operators. Again, looking at the photocopier, the operators required vary greatly in complexity; "placing paper" requires lifting the cover and

positioning a page, whereas pressing the “copy” button is a simple motor element. In the tasks used here, operators are all either single mouse clicks or keystrokes so that all actions are of more or less equivalent grain size.

Finally, we were interested in whether or not there are task characteristics which influence the frequency of post-completion error. In order to pursue this goal, two different tasks were used. Ideally, the two tasks should show different patters of post-completion error and thus provide some insight into the source of post-completion errors through examination of the differences between the two tasks.

To ensure that the tasks were new to participants, they were created for the experiment and presented as part of the science-fictional world of Star Trek. The two tasks used were the phaser task and the deflector task, each of which had two versions, a control version and a post-completion version. Control and post-completion versions of the task contained the same steps with the order of steps being the same until the end of the task where the post-completion version has the main goal satisfied before the task is completed.

The phaser task required the participants to carry out a fairly complex procedure involving several subgoals and a time-consuming tracking subtask. A diagram of the task structure for the post-completion version of the task is presented in Figure 2 in which ovals represent goals or subgoals and rectangles represent required actions. The key part of the task is the final step of turning off the tracking. In the post-completion version shown in the figure, this step occurs after the main goal of the procedure has been satisfied, when the participants are told that the Romulan ship has been destroyed. In the control version, participants were not told whether or not the goal was accomplished until after the last step had been completed.

The deflector task was a much simpler task involving fewer subgoals and fewer steps. The diagram for the post-completion version of this task appears in Figure 3. In the post-completion version the participants can fulfill the main goal before satisfying the reset switching subgoal, but in the control version of the task, the reset switching subgoal and its associated actions must be performed before the other subgoals.

Method

Participants. 28 undergraduates from the Georgia Institute of Technology participated for extra credit in a psychology course.

Materials. The materials for Experiments 1 and 2 consisted of paper instruction manuals and Apple Macintosh IIfx computers running HyperCard software.

Design. Participants were divided into two groups; in one group, complex post-

completion, the participants were given the control version of the deflector task and the post-completion version of the phaser task. In the simple post-completion group, participants were given the control version of the phaser task and the post-completion version of the deflector task. Dependent measures included the total number of errors and the number of post-completion errors made by each participant.

Procedure. When the participants entered the experiment, they were first familiarized with the use of a mouse, if necessary. Participants then read a sheet describing the experimental procedure and then were trained on each task. In the training, participants first read the instruction manual for the phaser task. They then performed two trials with the phaser task, during which they could refer to the manual. The manual was then taken from the participant. Training continued with the phaser task until the participant was able to complete four trials without error. Then training with the deflector task began. This used the same training procedure as for the phaser: participants read the manual for the task, tried the task twice with the manual at hand, and then trained to criterion without the manual. During training, the experiment program emitted a loud beep whenever an error was made, so that participants were aware that they had made an error.

Once training had been completed, participants began the test period and were presented with a randomly ordered series of twenty test trials, ten per task. During the test trials, the manuals were not available to the participants and no error feedback was given.

In order to encourage participants to work quickly, the display screen for each task contained a timer that noted the amount of time (in seconds) that had passed since the participant started that particular trial. When the participants completed the experiment, they were shown a screen containing their average time for the task.

Results

The participants made a considerable number of post-completion errors in the complex post-completion task (the phaser task). Despite the fact that the instruction manual explicitly instructs participants on the correct action at the point of the post-completion step, 93% of the participants (13 of 14) in this condition made the anticipated post-completion error on their initial training trial. This is a high error rate, considering that participants averaged only about 2.5 errors over the course of an average of 6.9 training trials. Despite the fact that errors are considered difficult to predict, a particular error is appearing here consistently.

In the test phase, participants continued to make post-completion errors and, more importantly, at a higher rate that would be predicted by a stochastic error hypothesis. If errors are equiprobable, as a stochastic hypothesis would suggest, any particular step should be

responsible for 1/11 or 9.1% of the errors made. In fact, post-completion errors accounted for almost 15% of the errors made by participants (13 out of 87 errors). This is reliably more that would be expected given the stochastic error hypothesis, $z = 1.96$, $p = 0.03$, one-tailed (an alpha level of .05 was used for all statistical tests). It is important to note that this a conservative test of the hypothesis. Because the phaser often missed the target and thus did not always destroy it, the first ten steps were performed more times than the post-completion step, and so there were more opportunities for participants to make errors on these steps than on the post-completion step. Thus, the expected proportion of errors on the post-completion step should actually be less than the 9.1% figure used. No similar test was possible for the deflector task because a total of only three errors were made on this task; this floor effect makes comparisons between the two conditions impractical.

Discussion of Experiment 1

Experiment 1 achieved its main goal of demonstrating that post-completion errors can be generated in the laboratory as well as in daily life. On this basis alone, the phaser task in Experiment 1 can be considered successful in demonstrating an experimental paradigm in which procedural errors are common enough that it may be possible to get information about their source. In addition, these tasks have some degree of ecological validity (as discussed by Sellen & Norman, 1992) because unusual distractions and/or fatigue were not necessary to get participants to produce errors.

However, errors in general, and post-completion errors in particular, were not observed with the simple deflector task. This lack of errors on the deflector task is consistent with other attempts to generate such errors in the laboratory (Polson, personal communication). Polson's experiment used a simple procedure in which participants were asked to press an additional button after dialing a telephone number, and failed to generate many post-completion errors, or even many errors at all. So in at least two laboratory settings with simple tasks, few post-completion errors were observed.

While demonstrating that post-completion errors can be generated in the laboratory with a suitably complex task, Experiment 1 also cast suspicion on the hypothesis that such errors are stochastic in nature. Post-completion errors occurred in the complex task used in Experiment 1 more often than a stochastic theory would predict. That is, if the same mechanism caused errors in all steps in a procedure, errors at all steps should have equal frequency—but this is not what was found. Experiment 1 provides critical evidence that post-completion errors are somehow different from other procedural errors. Given that simple tasks seem to have failed to generate post-completion errors in the laboratory, the success in the complex task suggests that task complexity may be a key variable in generating such errors.

Unfortunately, “task complexity” is at best a vague term. There are many things that may make a task seem complex: number of actions to perform, number of subgoals, difficulty of executing individual steps, amount of information to be managed etc. Furthermore, the naturalistic tasks in which post-completion errors seem to appear, such as photocopiers and ATMs, are not particularly complex tasks by any of these measures. However, these tasks can become difficult if some sort of external memory load is present. Differing working memory demands may provide a more useful way to differentiate tasks. The phaser task used in Experiment 1 has enough subgoals and decision points to generate a fair amount of working memory demand by itself, which is not the case for simple tasks. On the other hand, a laboratory situation typically forces participants to work on only one task at a time, creating a situation in which the only working memory demands are the demands of the task itself. This is not the case in real-world settings; people often engage in other activities, such as having conversations or planning other tasks, while using photocopiers and ATMs. These other activities generate demands on working memory, and so make a simple task more difficult. Thus, working memory demand seems at least a plausible source of explanation for post-completion errors.

Working memory load is a promising avenue for two other reasons: First, working memory load has been empirically and theoretically implicated in the production of errors by other researchers (e.g. Hitch, 1978; Anderson & Jeffries, 1985; Libiére, Anderson, & Reder, 1994), although this work has typically not investigated systematic errors. Second, a formal, that is a computational, model of the effects of working memory load exists, and this model uses a representation—production rules—that is well suited to modeling procedure execution. This model is CAPS, and the next section details a theory of post-completion error based on CAPS.

A Computational Theory of Post-Completion Error

A theory of post-completion errors inspired by the results of Experiment 1 will be developed here which postulates that post-completion errors stem from goal forgetting which in turn is caused by excessive working memory load—despite the presence of correct procedural knowledge. This more general theory will then be instantiated as a computational model in the context of CAPS (Collaborative Activation-based Production System; Just & Carpenter, 1992).

General Theory

Omitting post-completion task elements (either goals, operators, or both) can be thought of as a kind of forgetting; the person performing the task simply forgets to perform some portion of the task. Since it is typically the case that the person performing the task knows the correct procedure, the error’s source is not permanent or long-term memory (LTM) forgetting—the correct knowledge is present in long-term memory. In Experiment 1, for instance, participants

had to meet a strict learning criterion, so there is good reason to believe that they had the correct procedural knowledge. One could argue that errors are a result of a failure to retrieve the appropriate material from LTM, but this explanation does not clearly explain why errors in one part of the the task would be more common than errors in any other part, and why so many tasks with this structure would be prone to the same type of error. If the forgetting that causes post-completion error is not of information from long-term memory, working memory seems a likely place to search for an explanation, as suggested in the discussion of Experiment 1.

The central claim of the theory is that post-completion omissions can be explained in a relatively straightforward way as goal loss from working memory. The concept underlying this mechanism is simple: Goals in working memory supply activation to their subgoals, so such subgoals only receive activation as long as their parent goal is active. When a goal is eliminated from working memory by being satisfied, any subgoal of that goal which is still present in working memory loses the activation supply from the satisfied goal. When working memory load is high, the loss of support from a parent goal goal may lead to the loss of still-unsatisfied subgoals.

In the photocopier example, the main goal is to obtain copies. While this goal is active, it supplies activation to the necessary subgoals such as positioning the original document, and then later removing it. However, when the copies are actually picked up from the paper tray, the main goal of obtaining copies is satisfied and no longer maintained. Since the subgoal of retrieving the original depends on the presence of the main goal, the length of time that the main goal stays in working memory is critical. If the main goal falls below threshold too soon, then the subgoal will not receive enough activation to reach threshold. If, on the other hand, the main goal is present in working memory for some time, the subgoal will safely reach threshold and be carried out. A higher working memory load is assumed to be associated with faster decay or displacement of information from working memory (e.g. Just & Carpenter, 1992; Card, Moran, & Newell, 1983), so whether or not the subgoal of removing the original from the copier reaches threshold depends on working memory load.

This should be the case for all tasks that exhibit post-completion structure; higher working memory load should be associated with more post-completion errors. Of course, working memory load can come from a number of sources, such as the task itself (as in the phaser task in Experiment 1), or from other tasks done in conjunction with the post-completion task.

This CAPS-based account is similar to both the C-I account and the ATS account in that they are all based on goal activation. It is different from the other approaches because instead of relying on goal similarity, it relies on working memory activation. It may, of course, be possible

to integrate a working memory account and a similarity account; however, such an endeavor is likely to be profitable only to the extent that the two approaches are supported by empirical data. At this point, as previously noted, none of the accounts has much in the way of empirical support.

CAPS

The general mechanism outlined above has been instantiated as a CAPS model. The most comprehensive description and illustration of CAPS can be found in Just and Carpenter (1992). As with most production systems, CAPS has two memory systems, a long-term memory and a working memory. Working memory contains elements such as goals and propositions. In CAPS, each element in working memory has some continuous activation value associated with it. Not all working memory elements can be used in the matching of productions; an element's activation value must be above a threshold in order for that element to be matched.

The long-term memory contains productions, which are if-then rules. Productions match their "if" conditions against all the above-threshold elements in working memory. On a given execution cycle, all productions that match are allowed to fire in parallel; there is no conflict resolution. When a production fires, it can cause one of two things to happen: an action, such as pushing a button, or a request for activation of some element or elements in working memory. If a production requests activation for an element that is not present in working memory, that element is created.

In CAPS, there is a ceiling to the total amount of activation that may be present in the system at any given time; this ceiling is the model's working memory capacity. On a given cycle, all the productions that have their conditions matched request some amount of activation from the system. The amount requested is totaled and added to the total activation of the currently active working memory elements. If this total exceeds the ceiling, the system cuts back the amount of activation that is propagated and maintained according to the amount that total activation requested exceeds the ceiling. For example, if a set of productions fire requesting 60 units of activation and currently active elements use 60 units, then 120 total units of activation are requested. If the ceiling is 100 units, the activation propagated by each production and the activation of each working memory element will be cut back corresponding to the amount requested that is above the ceiling. In this case, that would be 10/120 (or 1/6) of the requested amount. That is, 5/6 (100 units) of the 120 units of activation will be propagated and maintained as reach individual element and production is scaled back to 5/6 of its request.

The primary implications of such a capacity limit are twofold. First, when the system is operating at the capacity ceiling and memory load is increased, the functional decay rate for elements in working memory will increase—that is, elements will be displaced faster. This is

because the amount of scaling back will have to increase to accommodate the increased demand. Second, it is not necessary to implement a goal stack in such a system. Stack-like behavior emerges from the differential activation levels of the goals that are present in working memory. Both of these features are critical in producing post-completion errors in the model.

Modeling

The phaser task from Experiment 1 was modeled in CAPS, in both the post-completion version and the control version. The procedural knowledge necessary to execute the procedure for the phaser task was initially specified as a GOMS (Goals, Operators, Methods, and Selection rules) model according to the guidelines found in Card, Moran, and Newell (1983) and Kieras (1988). GOMS-based task analyses have been shown to be an excellent description of the procedural knowledge required to operate such devices in their ability to predict both learning and execution times (Kieras & Polson, 1985; Bovair, Kieras, & Polson, 1990; Gong & Kieras, 1994). The GOMS model of the phaser task was transformed into a set of productions using a procedure similar to the one described in Bovair, Kieras and Polson (1990).

Goal management was handled by adding goal maintenance productions which have the following general form:

```
IF the parent goal [is above threshold, AND
      NOT the goal [is above threshold], AND
      the preconditions for the goal are met, AND
      the goal state has not been reached
THEN

      propagate activation to the goal
```

This yields behavior quite similar to a goal stack because while the parent goal is above threshold, a subgoal whose preconditions are met will receive activation until it is satisfied. However, unlike in a goal stack, goals can be matched after they are satisfied provided they still have enough activation. Satisfied goals will lose activation at a rate that depends on the load on working memory; with a high working memory load they lose activation and drop below threshold very quickly, but with a low working memory load they lose activation more slowly and may stay above threshold for some time.

The results of running the control and post-completion versions of the CAPS model are not the same, and depend on the value of the activation ceiling given to the CAPS system. If the activation ceiling is 7.0 units or higher, both models execute the task without error. However, if the ceiling is lowered, the post-completion step of turn off tracking is omitted by the post-

completion version of the model. That is, with a high enough working memory capacity, the error does not occur, and with low capacity the error is always made. This suggests a possible relationship between working memory capacity and post-completion error.

It is important to note that this mechanism does not necessarily suggest a simple correlation between individual differences in working memory capacity and post-completion errors. This mechanism is a threshold-driven mechanism, so if the task demands on working memory are very low, it is possible that everyone will be above threshold. That is, post-completion errors should be rare regardless of the capacity of the individuals involved. On the other hand, if the task in question places large demands on working memory, then post-completion errors should be common regardless of the capacity of the individuals involved since everyone will be below the threshold. According to the model, individual differences should only matter within particular ranges of task demand—when the demand places the threshold within the range of individual differences. Empirical tests will then be necessary to determine the memory demands actually caused by particular tasks for particular subject populations.

In summary, the CAPS model predicts that, even with the correct procedural knowledge, post-completion errors can occur as a result of high working memory load. Experiment 2 is intended to evaluate this prediction.

Experiment 2

Experiment 2 was designed with two purposes in mind: first, to demonstrate that the post-completion errors observed in Experiment 1 are not peculiar to the phaser task; and second, to test the predictions of the CAPS model. The first goal is served by using a new task in addition to the phaser task. While this is important in demonstrating the generality of the phenomenon, it does not help understand the cause of post-completion error. The CAPS model, however, does specify a mechanism by which post-completion error can arise: working memory load. The extent to which working memory is taxed is a function of two things: an individual's capacity and the load imposed by the task. Experiment 2 seeks to address these issues by extending Experiment 1 through the addition of working memory capacity assessment of participants and an external memory load condition.

The CAPS simulation model predicts that capacity and memory load could all interact to affect the likelihood of post-completion error. In general, high capacity and low memory load should produce fewer post-completion errors. Conversely participants with lower capacity in a memory intensive condition are expected to make more post-completion errors. Of course, it may be that any one of these factors is enough by itself to produce post-completion errors on the tasks used here.

Method

Participants. 64 undergraduates from the Georgia Institute of Technology participated in the experiment for extra credit in a psychology course.

Materials. The materials for Experiment 2 consisted of Apple Macintosh IIfx computers running HyperCard and paper instruction manuals.

Design. Experiment 2 was a three-factor between-subjects design consisting of the following three factors: working memory load, working memory capacity, and mix of tasks (described below). The first factor, working memory load, had two levels—load and no load—to which participants were randomly assigned. Participants in the no load condition performed the tasks normally, as in Experiment 1. While executing the tasks, participants in the load condition performed a concurrent, unrelated task similar to the one used by Reber and Kotovsky (1992). In this task, participants were presented with auditory stimuli (spoken letters) at the rate of approximately one letter per three seconds. At random intervals ranging from 9-45 seconds, participants heard a tone rather than a letter. When they heard this tone, participants were to recall the last three letters they heard by writing them down. Thus participants in this condition were forced to not only remember three letters, but to constantly update the letters that they were to remember. There were several reasons for using this particular manipulation; in particular, it has been shown to generate behavioral differences (Reber & Kotovsky, 1992), and it uses materials unrelated to the procedural tasks.

The second factor, working memory capacity, is a continuous individual difference variable. In order to obtain a good estimate of working memory capacity, two different tasks were used, and these tasks were intended to have as little domain-specific crossover as possible. A subject's working memory score was the average of their z-scores on two working memory tasks: Sentence Digit and Operation Word, taken from Turner and Engle (1989) and described below. These tasks make use of different items in their computation and storage components in order to minimize domain-dependence within each task. A good estimate of general working memory capacity should be provided by the convergence of these two measures.

In the Sentence Digit task, participants read a sentence followed by a digit from 1-9. They were asked a simple question about the sentence and told to retain the digit until recall was requested. One or more sentence-digit pairs were presented, after which participants were asked to recall the digits. Participants began with one sentence-digit pair, then received two pairs, and then three, and so on until the participant either incorrectly answered questions about the sentence or misremembered digits on two out of three trials for a given span. The subject's score was the total number of digits recalled correctly.

Operation Word was similar to Sentence Digit, except that instead of being presented with a sentence, participants were presented with an arithmetic problem along with a word. Participants were asked to perform the arithmetic computation and remember the word. The number of problem-word pairs was gradually increased until participants made errors in either computation or recall on two out of three trials. A subject's score was again measured by the total number of words the participant recalled correctly.

The final factor, task mix, had two levels to which participants were randomly assigned: phaser post-completion and transporter post-completion. As with Experiment 1, there were two tasks, each having a post-completion version and a control version; for Experiment 2 these tasks were the phaser task and the transporter task. Both versions of the phaser task were identical to those used in Experiment 1. The new task, the transporter, was identical to the phaser task in terms of the number of subtasks and operators required to complete the task, as well as overall task organization; the differences were primarily in the names of the controls and the appearance of the display. The phaser and transporter also contained tracking subtasks at identical places in the procedure, though the tracking tasks were not themselves identical. Participants in the phaser post-completion task mix group performed the control version of the transporter task and the post-completion version of the phaser task while participants in the transporter post-completion task mix group performed the control version of the phaser task and the post-completion version of the transporter task.

Number of errors was again the dependent measure in these tasks. Both post-completion errors and other errors were recorded, although post-completion errors are of primary interest here.

Procedure. The participants were run in two sessions, spaced one week apart. In the first session, participants read the instruction sheet describing the experimental procedure, after which the training procedure began. In the training, participants first read the instruction manual for the phaser task. They next performed one trial with the phaser, during which they were allowed to refer to the manual. This was essentially a familiarization trial. The manual was then taken from the participant to encourage memorization of the procedure. Training continued with the phaser task until the participant was able to complete three trials without error. During training, if the participant made an error executing the procedure, the trial was ended and the participant was told the correct action that was to be next executed. Once training with the phaser was finished, training with the transporter began. The training procedure was nearly identical to training with the phaser: participants read the manual for the task, tried once with the manual at hand, and then trained to criterion without the manual.

The second session consisted of two parts: working memory assessment and test trials of the two tasks. Participants first performed the working memory tasks until they reached the limits of their span. Once the participant had completed working memory assessment, they performed ten trials each with the phaser and transporter, presented in random order. That is, unlike in training, trials were not blocked but randomized to prevent differential practice effects. During the second session, the experiment program emitted a loud buzz whenever an error was made so that participants were aware that they had just made an error. Participants were, however, allowed to continue the trial. This feedback was provided to make the tasks more similar to naturalistic tasks; in everyday life, people are often given feedback about their success in executing routine tasks.

In order to encourage participants to work quickly, the display screen for each task contained a timer that displayed the amount of time (in seconds) that had passed since the participant started that particular trial. At the end of each trial participants were shown their average time to complete a trial for that task.

Participants were run alone or in pairs. In order to minimize the amount of noise interference between participants assigned to the working memory load condition, participants wore headphones connected to their Macintosh to present the letters for recall. To maintain consistency, all participants wore headphones regardless of whether they were run alone or in pairs.

Results

Data from three participants were excluded for being more than three standard deviations from the mean in total errors on the subject's first trial. These participants presumably forgot the task procedure and thus would not qualify as having correct procedural knowledge.

Corroborating the results of Experiment 1, post-completion errors accounted for a reliably higher proportion of errors (294 of 1,1178, or 25.0%) than would be predicted by a stochastic error theory (as described earlier, 9.1%; $z = 18.94$). This result was anticipated and necessary to establish the robustness of the phenomena. It is also important to establish the equality of the control conditions and secondary tasks. Participants who had the transporter as the control task made an average of 8.2 errors over the ten trials, while those who fired the phaser as their control task averaged 7.7 errors. This difference was not reliable, $t(59) = 0.33$, $p = 0.74$, with power of approximately 0.7 to detect a medium-large difference (Cohen, 1988). We can claim with reasonable confidence, then, that the two control conditions were equivalent in terms of error rate. For those participants who received the extra memory load—the letter recall task—there was no evidence that participants differentially traded off effort between that task and the procedural task. That is, scores on the letter recall were not better or worse for

participants concurrently doing the phaser task vs. those doing the transporter task, $t(28) = 1.32$, $p = 0.20$. Accuracy on the letter task was also not reliably correlated with error rate, either for post-completion errors or total errors ($r = 0.16$, $p = 0.39$ and $r = 0.25$, $p = 0.18$ respectively), so it appears that all participants spent roughly the same amount of effort on the secondary task.

The critical question, then, is were subjects more error-prone in the post-completion conditions and if so, was that difference exaggerated by working memory demand. To test this, the difference between the number of errors in the control condition and the post-completion condition was computed for each participant. Task mix, working memory load, working memory capacity and all the associated interactions were regressed on this difference score (type III sums of squares were used for all terms in the regression equation). As might be expected based on the CAPS model, there was a reliable interaction of memory capacity and memory load, beta = -2.1, $t(53) = -2.01$, $p < 0.05$. To make the interaction easier to interpret, the continuous working memory variable was split on the median to yield two groups, low capacity and high capacity. This interaction is depicted graphically in Figure 4; essentially, the increase in error rate between control and post-completion conditions is not reliably different from zero except for low-capacity subjects under memory load. Presumably, these participants were below the critical working memory threshold, which is consistent with the CAPS threshold account.

The capacity by load interaction was not, however, the only reliable effect. There was also a reliable effect of task mix, beta = 1.66, $t(53) = 2.08$, $p = 0.04$, which was not foreseen. For those participants who had the phaser task as the post-completion task, the mean increase in errors from control to post-completion was 1.4, whereas for those who had the transporter as the post-completion task, the mean increase in errors was 5.3. Since the tasks are roughly equivalent as control tasks and have the same structure, the first obvious difference between the tasks is the target tracking subtask performed before either firing the phaser or engaging the transporter. The original CAPS model did simulate performing the tracking task but did so for a fixed time interval. To see if the CAPS model was affected by tracking time, the model was modified so that the tracking subtask could run for a variable numbers of production cycles. As it turned out, different numbers of cycles spent on the tracking subtask have different effects on working memory, and thus on the likelihood of making the post-completion error.

The key reason for these differential effects is that the amount of time spent tracking the target affects the amount of activation requested in working memory.When working memory capacity is limited, scaling mechanisms cut back the amount of activation propagated by productions. This means that though the working memory elements required for the tracking task attempt to stabilize at some level above threshold, they take a few cycles to get there. While they are attempting to stabilize,the scaling mechanisms also gradually displace older working memory elements that are not being actively maintained. The longer the tracking time, the more

likely it is that these older elements will have been displaced before the phaser is fired, and so the less effect they have on the total capacity when the post-completion goal becomes relevant. This is functionally equivalent to a small increase in capacity at that point. When tracking times are short, then the older elements are less likely to be displaced, and the capacity available for the post-completion goal will be smaller. How much effect this will have depends on the total activation available. If there is a large enough amount, then there should be no effect of tracking time; it is only when there is limited activation available that tracking time will make a difference. The prediction that longer tracking times should lead to fewer errors is somewhat counter-intuitive, and is a function of the gradual displacement of irrelevant working memory items during the mostly perceptual-motor tracking task. This type of tracking task is not normally found in the naturalistic tasks in which post-completion errors are common, so this prediction is peculiar to the tasks used in Experiment 2 or other tasks that contain a mostly motor subtask of variable length just before the post-completion subgoal.

The data bear out that the tracking times for the two tasks did indeed differ. Average tracking time for the phaser task was 16.45 seconds, which is reliably higher than the mean tracking time for the transporter task, 7.76 seconds, $t(60) = 6.99$, $p < 0.001$). The shorter tracking time for the transporter task leads to the prediction that more post-completion errors should occur with the transporter task. This is consistent with the difference that was observed; a larger increase in errors was associated with the transporter task, as described above.

The next obvious question to be answered by the data is whether the observed increases in error rate associated with the post-completion condition are actually due to post-completion errors themselves. In answer to this question, the regression of post-completion errors on task mix, working memory load, and working memory capacity reveals a reliable three-way interaction of the factors, beta = -0.30, $t(53) = -2.40$, $p = 0.02$. To make the three-way interaction simpler to interpret, the continuous working memory variable was again split on the median to yield two groups, low capacity and high capacity. This allows the data to be analyzed by factorial ANOVA, which yielded the same three-way interaction between task mix, working memory load, and working memory capacity, $F(1,53) = 5.72$, $p = 0.02$. The reliable three-way interaction reflects the influence of working memory load on post-completion errors in that higher load is associated with more post-completion errors and less load with fewer errors. Mean number of post-completion errors as a function of memory load and memory capacity for the phaser task are presented in Figure 5a and for the transporter task in Figure 5b. These data also support the idea of a working memory-based threshold; the only participants that appear to be below the threshold are the high capacity participants executing the phaser task without an external load, who made post-completion errors 15% of the time (i.e. a mean of 1.5 errors over ten trials). All other groups were within one standard error of making the post-completion error

at least half the time. Most notably, the group with the smallest available capacity, the low-capacity participants executing the transporter task while working with a concurrent memory load, made post-completion errors more than 75% of the time. Post-hoc tests confirmed that the highest-error cell and the lowest-error cell are reliably different and the remaining six cells are statistically indistinguishable. It should also be noted that the same pattern did not exist for other (i.e. non post-completion) errors. In particular, there was no effect for task mix nor did task mix interact with any of the other factors for other errors.

Another important question concerns the possible effects of practice: do participants learn not to make the post-completion error over time? As can be seen in Figure 6, the answer to this question is not entirely clear. While the number of errors did decrease over the first five trials ($F(4,212) = 3.54$, $p < .01$), performance was quite stable over the last five ($F(4,212) = .023$, $p = 0.92$). So there did seem to be some initial learning as participants apparently tried not to make the error. However, the robustness of the error is demonstrated by the fact that subjects were not particularly successful at eliminating it, as over 40% of the participants still made the post-completion error on the tenth trial.

Discussion of Experiment 2

The results of the two experiments are consistent with the CAPS model of the phenomena. Under conditions in which working memory capacity was not taxed, such as the deflector task in Experiment 1 and the high-capacity participants executing the phaser task without external load, post-completion errors were extremely rare. On the other hand, when working memory demands were increased with a more complex task, shorter tracking time, external load or low-capacity participants, the error was much more common. This is what would be expected based on the CAPS model: when enough capacity was available, the model did not make the error, but with less available capacity it did. The CAPS model predicts the occurrence of a particular error under relatively well-specified conditions and the empirical results support those predictions. Thus, not only does the CAPS model provide a theoretical account of a common, everyday phenomenon, it makes predictions about when the error will be observed that are consistent with laboratory data.

One thing that makes the CAPS explanation appealing is that no special modifications were needed to make the CAPS model produce post-completion errors. Productions were written with uniform strengths and none of the many parameters available to the CAPS programmer were manipulated. A simple GOMS model of the procedure was translated into CAPS productions using style rules established for other models, and the error emerged naturally from the CAPS memory activation system. The productions used contain the correct knowledge to perform the task and are capable of producing error-free behavior, but the model is

capable of making post-completion errors with no special error or noise generator. In addition, the CAPS model was able to account for the counter-intuitive dif ference associated with the different tracking times for the two tasks, despite the fact that this factor was not foreseen when the model was initially constructed. Certainly, this is less than optimal in that it was a post-hoc fit, but the fit required only allowing a formerly fixed parameter to vary—no changes in approach or mechanism were required. Thus, the CAPS model presented here is in accord with Sellen and Norman's (1992, p. 318) call for "a shift away from viewing errors as special events or as the product of special error-producing mechanisms and instead toward a view that characterizes errors as the normal by-products the design of the human action system."

This is not to say that the CAPS model is a model of errors in general. While the CAPS model both makes and provides an explanation of post-completion errors, it neither makes nor explains the other errors made by participants in these experiments. It may well be that other errors are in some way related to working memory failures, but it would be difficult to claim that they are the result of exactly the same mechanism, since errors did not occur with uniform frequency. Nor does this account rule out other accounts of post-completion error—there is certainly still promise in the C-I and goal stack accounts. Since the other explanation do not involve working memory capacity limitations, however, they would have a difficult time accounting for the present findings.

The issue of learning in these situations is still relatively unexplored. While there is certainly evidence for learning found Experiment 2, it is not clear whether people can learn to stop making post-completion errors. In this experiment, participants were made aware that they had made an error by a loud buzz, but there was no further feedback given to them. In more naturalistic settings, people may be more motivated to learn not to make the error because the cost of making it can be much higher. If this is the case, then the implications for the CAPS model are not clear, particularly since CAPS does not include any kind of learning mechanism. On the other hand, the "all or nothing" nature of the goal killoff and goal stack approaches makes it difficult to see exactly how the current data relate to these approaches.

General Discussion

The CAPS model described here provides an account of an everyday phenomenon which previously had not been well-understood. The relative success of this model in predicting post-completion errors has implications for a number of areas including the study of error, computational modeling, and human-machine systems design.

The study of systematic procedural error is not particularly advanced compared to the study of many other cognitive phenomena such as categorization or lexical decision. One reason

for this is that there is a shortage of laboratory data on systematic error, particularly when the errors are not knowledge-based. Part of the problem may be that errors are not always regarded as a phenomenon in their own right, as argued extensively by Senders and Moray (1991), but rather as some kind of stochastic indicator of other variables. Even when errors are considered as an important phenomenon, laboratory studies in this area are difficult to do perhaps because participants are typically not very obliging in producing particular types of errors in the laboratory in a reliable way. This may be because the laboratory situation tends to make people self-conscious and engage in self-monitoring more than they would otherwise. To generate systematic errors in the lab that mirror naturalistic errors, a successful approach may be similar to the one taken here: to use engaging game-like tasks that engage participants' interest making the task seem less artificial, and to use more complex variants of the simple, naturalistic tasks.

Although we have provided an account of one systematic error, this is not to say that all errors are systematic—many of them probably are stochastic. However, this work shows that it is a mistake to assume that all errors are stochastic and to treat them as a result or measure of some other construct. It may be more productive to make the opposite assumption that all errors are systematic and then attempt to determine the source of each kind of error. Though this approach may be time-consuming, we stand to learn a great deal about how people execute procedures through examining how people fail to execute them correctly. A great deal has been learned about human memory by examining the kinds of errors found in recognition and recall; it may be possible to learn an equal amount about how people learn and execute procedures by more closely examining the errors that are made in procedural tasks.

Another reason for the shortage of work on systematic procedural error may be a lack of theoretical guidance; current theories of error are not specific enough to make precise predictions. This work demonstrates that with a clear and precise enough theoretical tool, a computational model, it is possible to generate reasonable predictions about a systematic procedural error. It is important to note, however, that simply having a computational model is not enough—one needs to have the right model. The choice of CAPS as a modeling tool was critical in specifying the hypothesis that post-completion is a function of working memory load, because CAPS is one of the few systems available in which the concept of limited working memory capacity is operationalized. In addition, the limits on working memory capacity allow CAPS to make errors and this ability is an essential aspect of the CAPS explanation for many important phenomena. This is in contrast to many other systems which are designed to model only the correct acquisition and execution of procedures, and are therefore handicapped in their ability to explain the error behavior that is an important aspect of how people interact with their world. Errors are important both because they are a part of human behavior and because examination of how and why people fail can potentially provide key information about how the

cognitive system operates.

Though CAPS is well-suited for modeling post-completion errors, CAPS is not without its problems as a cognitive model. Because CAPS does not simply delete goals from working memory when they are satisfied, it is important that satisfied goals get displaced so that they do not generate inappropriate actions. To execute a procedure correctly, goals must be executed in a specific order, and this need for correct ordering of goals makes it necessary for CAPS to be run at values of the activation ceiling low enough that displacement is forced. CAPS only loses things from working memory if it is trying to use more activation than it is allowed. If the system is not run with a capacity low enough to cause displacement, satisfied goals do not decay and correct sequencing is not ensured since irrelevant goals may be active. That is, CAPS can only work correctly if there is some strain on working memory—yet people seem to work quite effectively when there is no such strain. There are approaches that solve this problem which are not supported by CAPS. For example, one approach, for which there is considerable evidence (for a review, see Salthouse & Babcock, 1991) is to have the activation of all elements in working memory that are not explicitly maintained decay over time. If a satisfied goal was no longer explicitly maintained, goals would decay after being satisfied regardless of capacity limitation. This would yield a model that should run correctly without having to strain working memory. As far as we know, there are no computational models that have a decay-based working memory system. More research is clearly needed in this area to determine the best way to construct such a model.

The CAPS model is lacking in other areas as well. While the CAPS model predicts post-completion errors reasonably well, it makes no other errors while executing the tasks. The participants, on the other hand, averaged one non-post-completion error approximately every other trial. The CAPS model is silent on the possible causes of these other errors. It may be possible to model some or all of the non-post-completion errors made with CAPS, but they do not simply emerge the way that post-completion errors did in our model.

In addition, the number of errors participants made seemed to decline slightly with practice. Since CAPS does not include a learning mechanism, it cannot account for this trend. It may be possible to account for this kind of small improvement over time with a strength-adjustment mechanism such as the one found in ACT or in some PDP models, but until such a learning mechanism is actually implemented inability to account for the reduction in errors over time can be considered a shortcoming of the model.

This is not unusual in computational modeling—certain systems are better at some things than others. For example, both Soar and C-I contain problem-solving and learning components that allow them to make predictions about important other aspects of simple naturalistic tasks,

such as use of ATMs, when the correct knowledge is not already present. (For an extensive review and comparison of Soar and C-I on ATM use, see Wharton, 1994).

Finally, this research has implications for design of human-machine systems as well. Designers of such systems would do well to avoid building post-completion structure into the interaction, even for tasks that may not themselves impose much of a working memory load. A person's working memory can easily become filled with task-irrelevant information at any time by daydreaming, planning the route to drive after getting gas, worrying about their account balance, and so on. Since the interface designer has no control over memory load, post-completion steps should always be avoided. There are examples of safeguards in real systems today, such as ATMs that will not dispense money until the card has been removed, ATMs where the card is swiped rather than inserted, and gas caps that are attached to the car so that they cannot be left behind. Of course, the suggestion to simply avoid the kinds of task structures that are prone to post-completion is hardly new—it is the same advice offered in the "cognitive walkthrough" approach developed by Polson and colleagues (Polson et al., 1992) and in some human-computer interaction textbooks (e.g. Dix, Finlay, Abowd, & Beale, 1993).

References

Anderson, J. R. (1983) The architecture of cognition. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1993). Rules of the mind. Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R., & Jeffries, R. (1985). Novice LISP errors: Undetected losses of information from working memory. Human-Computer Interaction, 22, 403-423.

Baars, B. J. (1992a). Experimental slips and human error: Exploring the architecture of volition. New York: Plenum Press.

Baars, B. J. (1992b). The many uses of error: Twelve steps to a unified framework. In B. J. Baars (Ed.), Experimental slips and human error: Exploring the architecture of volition. New York: Plenum Press.

Brown, J. S., & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. Cognitive Science, 4, 397-426.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text editing skill: A cognitive complexity analysis. Human-Computer Interaction, 5, 1-48.

Card, S., Moran, T., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Erlbaum.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences. Hillsdale, NJ: Erlbaum.

Dix, A., Finlay, J., Abowd, G., Beale, R. (1993). Human-computer interaction. New York: Prentice Hall.

Gong, R., & Kieras, D. (1994). A validation of the GOMS model methodology in the development of a specialized, commercial software application. In CHI'94 Conference Proceedings. Reading, MA: Addison Wesley.

Hitch, G. J. (1978). The role of short-term working memory in mental arithmetic. Cognitive Psychology, 10, 302-323.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. Psychological Review, 99, 123-148.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), Handbook of human-computer interaction. Amsterdam: North-Holland Elesevier.

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 365-394.

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

Kitajima, M., & Polson, P. (1992). A computational model of skilled use of a graphical user interface. In CHI'92 Conference Proceedings. Reading, MA: Addison Wesley.

Laird, J. E., Newell, A., Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. Artificial Intelligence, 33, 1-64.

Libiére, C., Anderson, J. R., Reder, L. M. (1994). Error modeling in the ACT-R production system. In A. Ram & K. Eiselt (Eds.), Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum.

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

Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Harvard University Press.

Norman, D. A. (1980). Errors in Human Performance. Technical report 8004, Center for Human Information Processing. University of California, San Diego. La Jolla, CA.

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

Polson, P. G., Lewis, C., Reiman, J., & Wharton, C. (1992). Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. International Journal of Man-machine Studies, 36, 741-773.

Reason, J. T. (1990). Human error. New York: Cambridge University Press.

Reber, P. J., & Kotovsky, K. (1992). Learning and problem solving under a memory Load. In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum.

Salthouse, T. A., & Babcock, R. L. (1991). Decomposing adult age dif ferences in working memory. Developmental Psychology, 27, 763-776.

Sellen, A. J., & Norman, D. A. (1992). The psychology of slips. In B. J. Baars (Ed.), Experimental slips and human error: Exploring the architecture of volition. New York: Plenum Press.

Senders, J. W., & Moray, N. P. (1991). Human error: Cause, prediction, and reduction. Hillsdale, NJ: Lawrence Erlbaum.

Turner, M. L., & Engle, R. W. (1989). Is working memory capacity task dependent? Journal of Memory and Language, 28, 127-154.

Wharton, C. (1994). A comparative study of Soar and the Construction-Integration model. Unpublished doctoral dissertation, University of Colorado.

Young, R. M., Barnard, P., Simon, T., & Whittington, J. (1989). How would your favorite user model cope with these scenarios? ACM SIGCHI Bulletin, 20, 51-55.

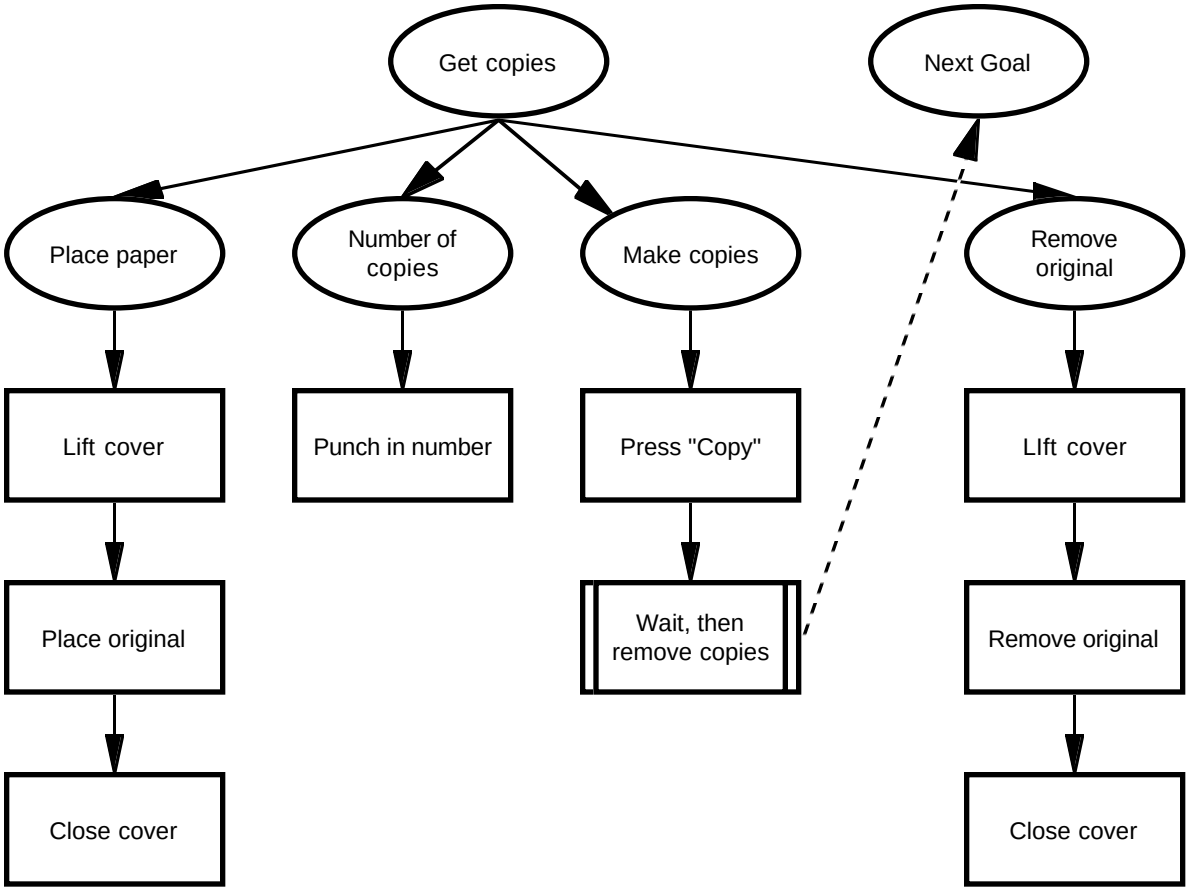


Figure 1. Task structure for the photocopier task.

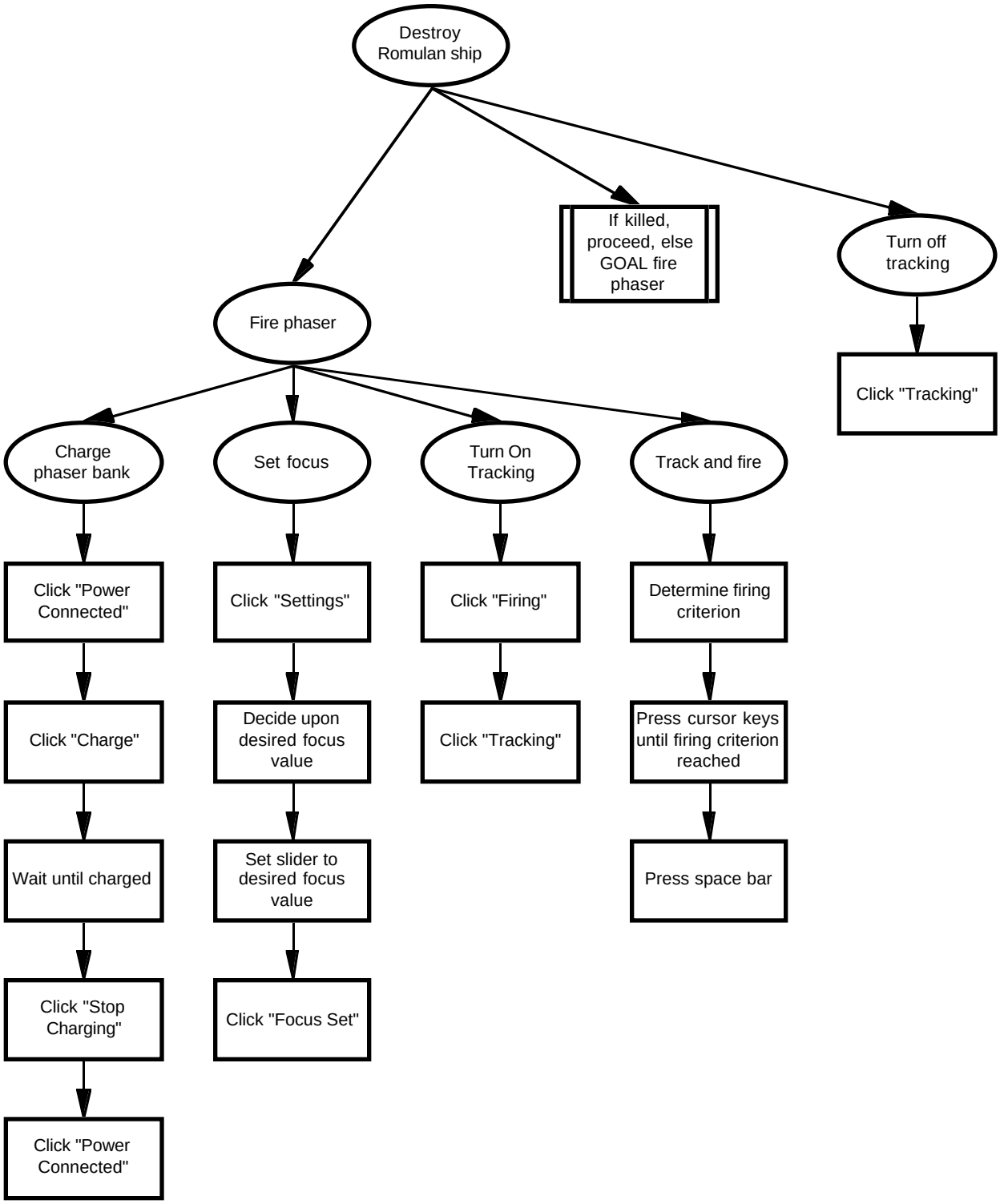


Figure 2. Task structure for the post-completion version of the phaser task used in Experiments 1 and 2.

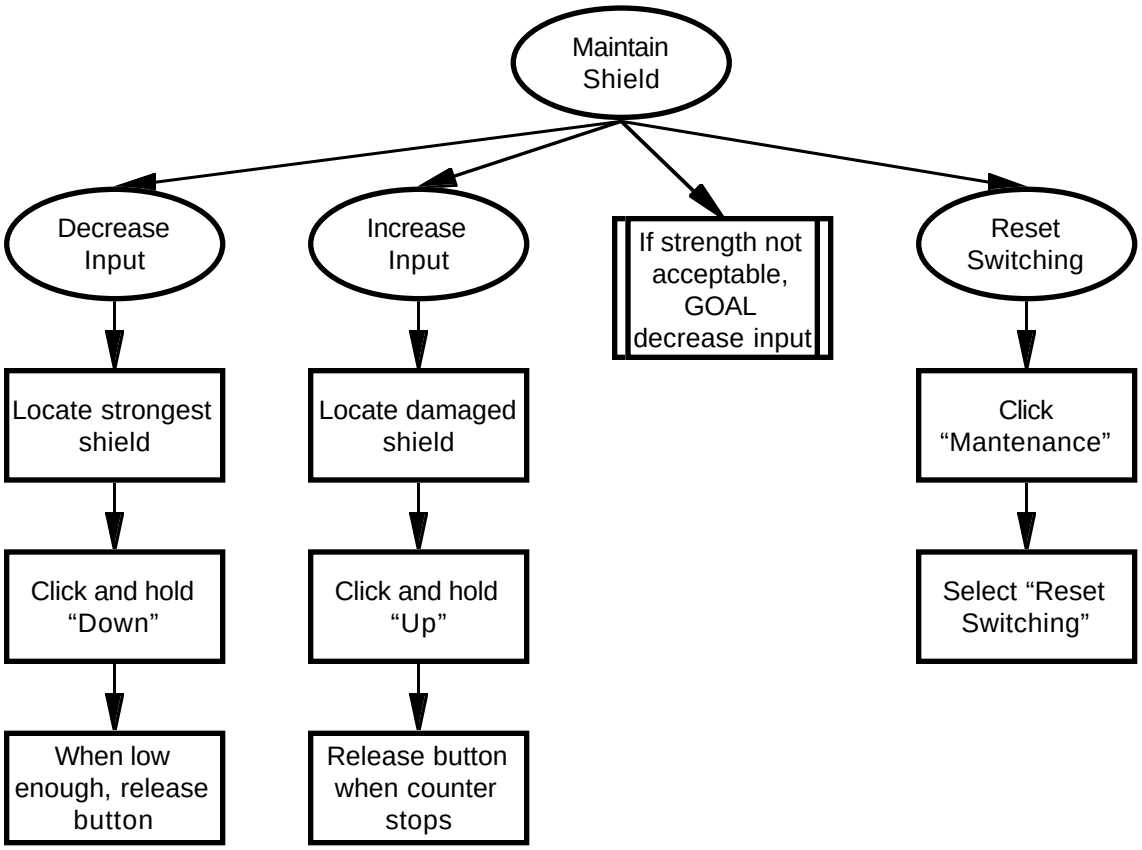


Figure 3. Task structure for the post-completion version of the deflector task used in Experiment 1.

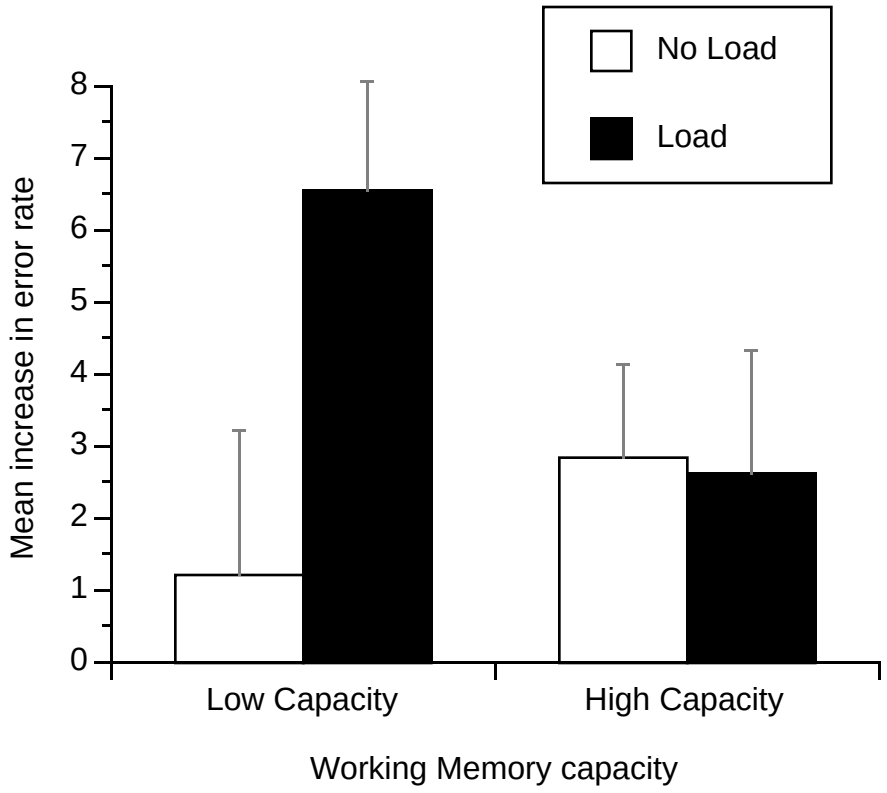
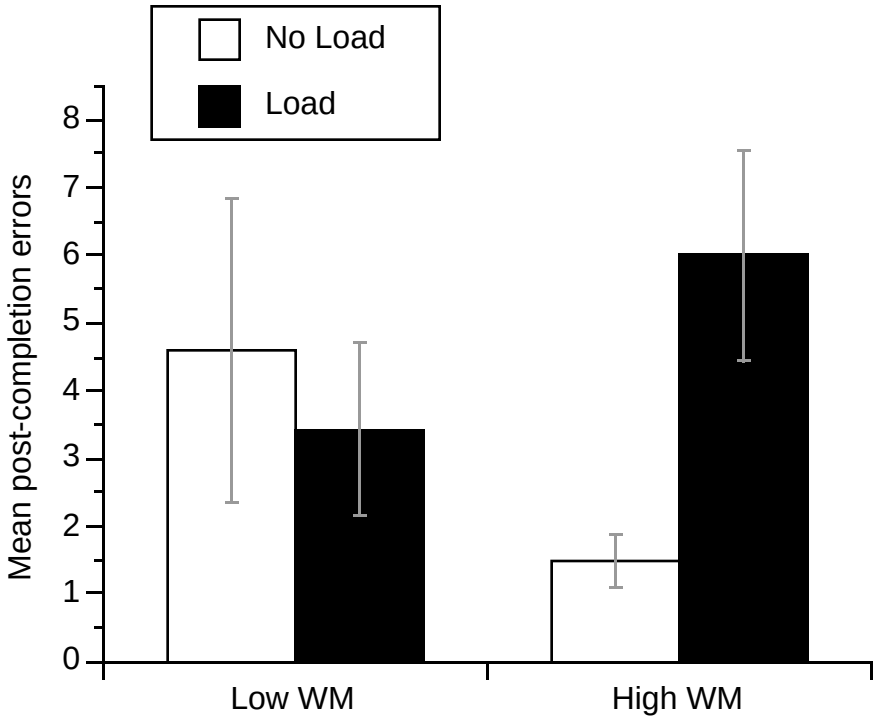
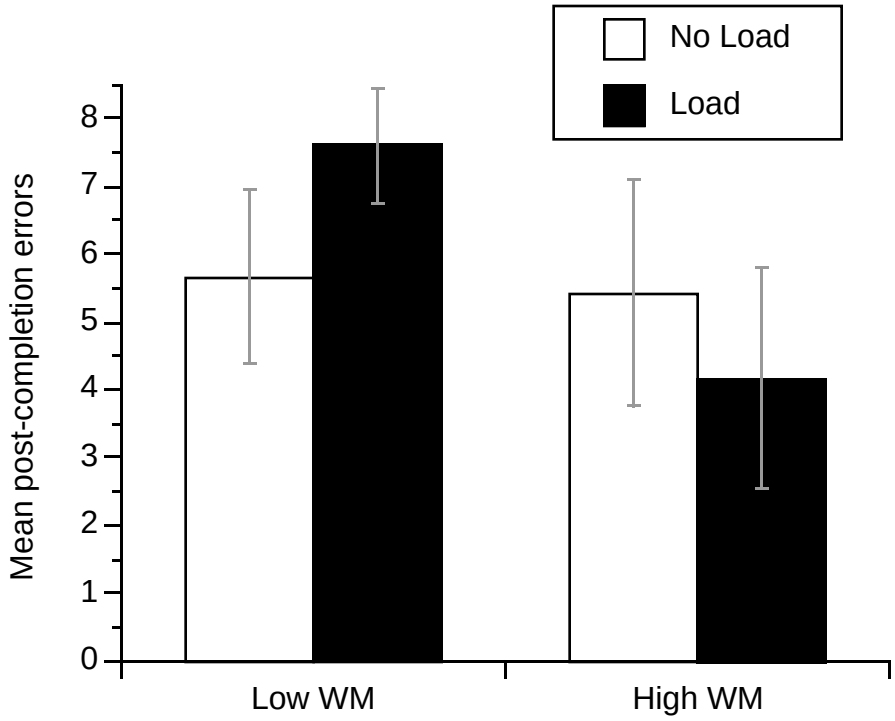


Figure 4. Mean increase in error rate from control to post-completion (+SE) as a function of memory load and memory capacity.

Figure 5. (following page) Mean number of post-completion errors (±SE) as a function of memory load and memory capacity for the phaser task (a) and for the transporter task (b) in Experiment 2.


No Load
Load
Mean post-completion errors
0
1
2
3
4
5
6
7
8
Low WM
High WM

(a)


No Load
Load
Mean post-completion errors
0
1
2
3
4
5
6
7
8
Low WM
High WM

(b)

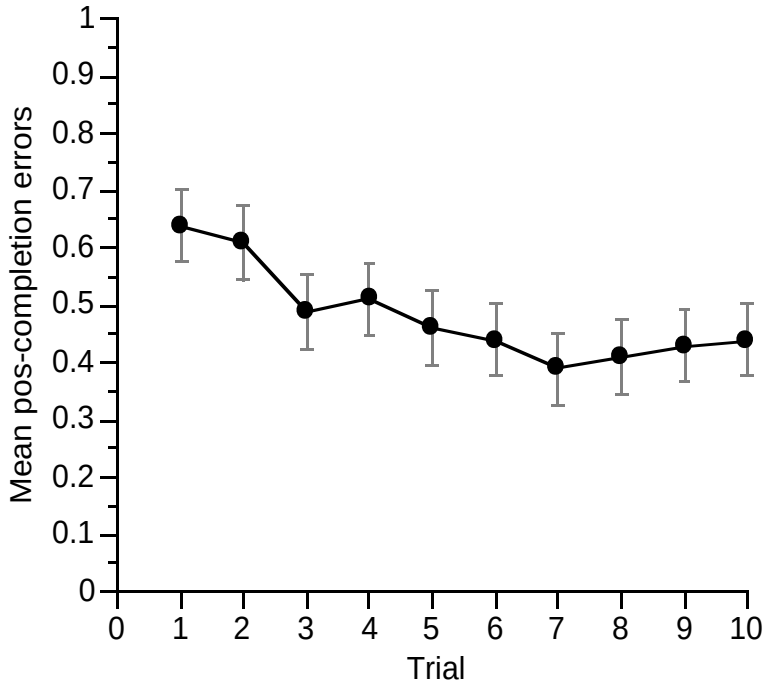


Figure 6. Mean number of post-completion errors (±SE) for all participants as a function of trial number in Experiment 2.