RICE UNIVERSITY

A Computational Model of Routine Procedural Memory

by

Franklin Patrick Tamborello, II

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Michael D. Byrne, Associate Professor,
Psychology and Computer Science

Philip T. Kortum, Professor in the Practice,
Psychology

David M. Lane, Associate Professor,
Psychology, Statistics, and Management

James L. Dannemiller, Lynette S. Autry Professor
Psychology

H. Albert Napier, Professor
Management and Psychology

HOUSTON, TEXAS
MAY 2009

ABSTRACT

A Computational Model of Routine Procedural Memory

by

Franklin Patrick Tamborello, II

Cooper and Shallice (2000) implemented a computational version of the Norman and Shallice's (1986) Contention Scheduling Model (CSM). The CSM is a hierarchically organized network of action schemas and goals. Botvinick and Plaut (2004) instead took a connectionist approach to modeling routine procedural behavior. They argued in favor of holistic, distributed representation of learned step co-occurrence associations. Two experiments found that people can adapt routine procedural behavior to changing circumstances quite readily and that other factors besides statistical co-occurrence can have influence on action selection. A CSM-inspired ACT-R model of the two experiments is the first to postdict differential error rates across multiple between-subjects conditions and trial types. Results from the behavioral and modeling studies favor a CSM-like theory of human routine procedural memory that uses discrete, hierarchically-organized goal and action representations that are adaptable to new but similar procedures.

ACKNOWLEDGMENTS

I wish to acknowledge my advisor, Mike Byrne, for all his mentoring not only during my dissertation project, but throughout all my years as a graduate student. Thank you, Carissa Chang and Adam Purtee for your assistance with data collection. Thank you also to Rick Cooper and Jay McClelland for your helpful comments. Thank you, Lynsey, my wife, Mom, Dad, Angy, and Melissa, my family, and to my friends for your patience, moral support, and love. I could not have done it without you all.

This work was funded by Office of Naval Research grants #N00014-03-1-0094 and #N00014-06-1-0056. The views and conclusions expressed are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the ONR, the U.S. government, or any other organization.

TABLE OF CONTENTS

LIST OF TABLES

1. INTRODUCTION

Humans regularly engage in complicated tasks composed of many steps. How do we know which step to perform when its time comes? How are we so good at performing these tasks quickly, and typically without error, once they have been learned and routinized? Once a task has been learned and becomes routine, why do we still occasionally fail in our performance? I intend to discuss several theories of human action selection in the literature and how they account for normal performance and error. I will include discussion of some of our own recent data and how it relates to the above issues.

Discussion of the relevant literature begins with the Norman and Shallice (1986) contention scheduling model and its instantiation as a formal computational model by Cooper and Shallice (2000). Next Botvinick and Plaut's (2004) simple recurrent network (SRN) computational model of action selection will offer a contrasting view of action selection and task sequence representation. The literature review continues with other theories concerned with action selection, namely GOMS and production system theories of human cognition, using ACT-R as an example. Discussion will conclude with some recent data from our own lab.

This dissertation's aim will be to answer some questions concerning human representation of routine action left unresolved by the literature. Are task memory structures organized simply by their statistical properties, as the SRN claims, or might some of the other cognitive and perceptual-motor process play a role in task representation as well? What happens when the routine procedure changes? Why do people err in routine procedures and how do they recover from those errors to finish the task?

The dissertation includes two experiments and a computational model of each of those experiments. The first experiment's aim was to determine whether people might delineate portions of a task by means other than learning which steps tend to occur together and which do not. Such means may include semantic factors such as semantic relatedness of step names. The second experiment was designed to delineate subtasks by statistical co-occurrence, but then destroy one subtask grouping by re-ordering its steps.

The two computational model accounts in the literature, one a contention scheduling model (CSM, Cooper & Shallice, 2000), the other a simple recurrent network (SRN, Botvinick and Plaut, 2004), each embody a different approach to human routine procedure representation. Whereas the CSM posits localist representations arranged into a hierarchy mirroring the task, the SRN claims distributed, holistic representation. The former is flexible in how old actions learned can transfer to new, similar tasks while the latter uses a statistical learning process to delineate subsequences of actions that may transfer.

Neither model specifies exactly how it performs many of the ancillary cognitive functions that go into producing actions, such as visual perception, recollection of facts, or manipulating the environment. The CSM, however, does at least specify how it may interact with other such cognitive systems to produce a wide range of human behavior. The SRN makes little effort at cognitive integration. As it is likely that other cognitive factors, like visual perception, have some role to play in human action production, it is desirable to instantiate a theory of action selection within a framework that can support those other cognitive processes. Therefore, since the CSM is amenable to working with a

generalized cognitive framework it will be mated to ACT-R so that interaction with other cognitive, perceptual, and motor processes may be examined together.

1.1. The Contention Scheduling Theory of Human Action Selection

Norman and Shallice (1986) put forward their contention scheduling model of skilled behavior as a symbolic interactive activation network. CSM claimed that people represent actions as schemas, which are associated representations of: trigger conditions, the actions themselves, how to perform the actions, any sub-schemas for any sub-steps of the procedure, and completion conditions. Norman and Shallice use the example of driving a car to illustrate schema interaction:

> …when the source schema for a task such as driving an automobile has been selected, all its component schemas become activated, including schemas for such acts as steering, stopping, accelerating, slowing, overtaking, and turning. Each of these component schemas in turn acts as a source schema, activating its own component schemas (braking, changing gear, signaling, and so on). (p. 6)

Together these representations form a strand of autonomous and self-sufficient processing structures, a "horizontal thread," that can generally carry out a well-learned task without need for attentional intervention (see Figure 1). While the component schemas might specify actions at an intermediate level (e.g., "change from second gear to third gear"), they leave the details associated with the lower-level actions entailed by the intermediate actions (e.g., "reach for gear lever") to other psychological processing structures, such as manual motor skills.

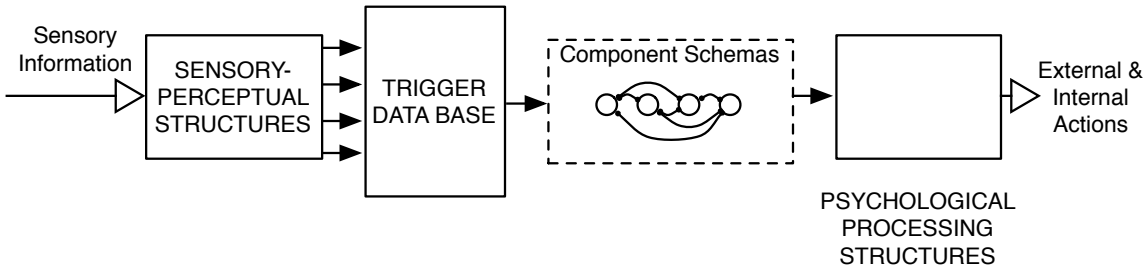

Figure 1. Horizontal thread from Norman & Shallice's (1986) contention scheduling model of human action selection.

Scheduling of action, and thus selecting from competing, similar, schemas is handled by contention scheduling. The sets of potential source schemas compete with one another in the determination of their activation value, then selection takes place on the basis of activation value alone. Competition is effected through lateral activation and inhibition among activated schemas. Importantly, the contention scheduling theory of action selection posits that "attention" is just another source of schema activation or inhibition originating in higher-order cognitive processes. These processes are involved in things like motivation and directed, effortful types of cognition like problem solving and selection of non-routine actions. Attention is not inherent to routine action selection. Nor is attention required to monitor actions in progress, but it can influence action selection by being a source of activation and inhibition input into the contention scheduling process from the Supervisory Attention System (SAS) – and that is its sole means of influencing the action selection process. Activation and inhibition from the SAS thus forms a "vertical thread" of influence tied to all schemas in the contention scheduling system. Thus the combination of horizontal and vertical threads enables a major feature of Norman and Shallice's contention scheduling theory: two levels of control, deliberate conscious control and automatic contention scheduling (Figure 2).

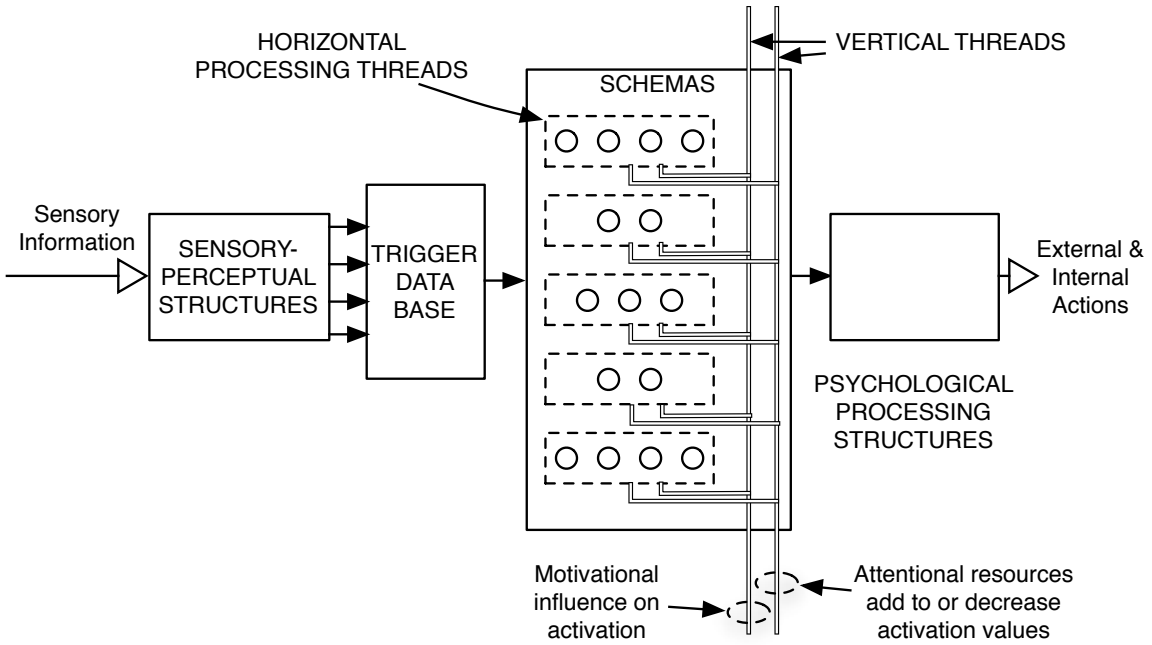

Figure 2. The overall system: Vertical and horizontal threads. When attention to particular tasks is required, vertical thread activation comes into play. Attention operates upon schemas only through manipulation of activation values, increasing the values for desired schemas, decreasing (inhibiting) the values for undesired ones. Motivational variables are assumed to play a similar role in the control of activation, but working over longer time periods. To emphasize that several tasks are usually active, with the individual components of each task either being simultaneous or overlapping in time, this figure shows five different horizontal threads. Some means of selecting the individual schemas at appropriate times while providing some form of conflict resolution becomes necessary. The interactions among the various horizontal threads needed for this purpose are indicated by the lines that interconnect schemas from different threads.

This feature is important for a number of reasons, one being that deliberate conscious control is often too slow to allow for the speed of many of our skilled behaviors. Additionally, trigger conditions can sometimes overwhelm influences from the SAS, as in the Stroop task. Perhaps most importantly, Norman and Shallice argue, is that certain types of errors seem to indicate that deliberate conscious control is not always required for action. In particular, “capture errors” occur when a person begins one task, and through inattention and/or distraction switches, before completion of the original task, to a new task that is at least as familiar as the original task. Reason and Mycielska (1982) documented an example of a capture error in a diary study they conducted. When passing

through his back porch on the way to get his car out, a subject stopped to put on his Wellington boots and gardening jacket as if to work in the garden. The new task captures the action selection of the person. Norman and Shallice argue that because attention, as far as action selection is concerned, is just another source of activation for contention scheduling, if it falls off at the wrong moment, activation from other sources select the wrong schema. According to Norman and Shallice, the Reason and Mycielska example reiterates the important point that the SAS is a separate system from the contention scheduling system that it oversees.

Cooper and Shallice (2000) instantiated the contention scheduling theory of control of routine actions as a computational cognitive model. They identified three levels of action at which humans operate: The lower level is mainly composed of the biomechanics of movements and the physical properties of targets, such as reaching and grasping. The intermediate level is composed of collections of lower-level actions, the actions at this level are completely specified, and they are specified to accomplish one and only one goal, such as making a cup of coffee. The higher level is composed of scripts and "memory organization packets" – typically groups of subgoals capable of being paused, interrupted, and resumed, to carry out some routine procedure. Higher levels of action control might direct activities such as going to a restaurant or visiting a doctor's office. It is at the intermediate level of action that Cooper and Shallice aimed their instantiation of the contention scheduling theory of routine action selection.

Cooper and Shallice's (2000) goal was twofold: first, to demonstrate the contention scheduling's viability as a theory of action selection in neurologically-intact individuals, and second, to further validate it as a theory of human action selection in routine tasks by

lesioning it and comparing its output to the behavior of two types of action disorganization syndrome patients reported by Schwartz et al. (1991, 1995, & 1998). The task Cooper and Shallice modeled was the same preparation of a cup of instant coffee observed by Schwartz et al.

The coffee preparation task is a relatively simple, seven-step procedure one might follow in the preparation of a typical cup of instant coffee. Cooper and Shallice (2000) conceptualized it as having a hierarchical structure with one superordinate goal (prepare instant coffee), three subgoals (sugar into coffee, milk into coffee, and grinds into coffee), and two equivalent, alternative methods for two of its subgoals (add sugar from packet/ bowl, add coffee from jar/packet).

For purposes of modeling, the hospital room environment in which Schwartz et al.'s patients were observed while making coffee was abstracted in Cooper and Shallice's (2000) study to be an 8 x 4 grid representing just the tray upon which coffee-making materials and implements would have been placed. Objects had features representing contents (for packets and containers), state (open or closed), and position. Schema selection is largely driven by the representation of the environment and, at the lower action level, by ordering constraints that are specified as symbolic preconditions. For example, if "add sugar" has been selected as the active subgoal, the only action possible, given the described state of the environment, will be "pick up spoon." Schemas that may achieve this goal receive excitation from the environment – schemas specify arguments, or what objects they may operate on or with (e.g., a spoon must be in hand in order to stir). Once an object has been picked up (normally, a spoon), symbolic preconditions prevent top-down excitation from immediately triggering inappropriate actions, such as

"put down." Instead, environmental triggering biases selection toward "dip spoon," which is selected and then inhibited. Similar processes then lead to the selection and inhibition of "empty spoon," which fulfills the precondition of "put down," and allows top-down excitation to trigger that schema. The "add sugar" subgoal is then completed, and the environment triggers another subgoal, perhaps "add milk." Additionally, because two schemas may potentially receive the same amount of total excitation, noise is added to each schema's activation to prevent ties and to lend some degree of stochasticity to the model's behavior.

1.1.1. Modeling Human Error in Contention Scheduling

The intermediate level of action in which Cooper and Shallice's (2000) contention scheduling model operates corresponds closely with the rule-based level of human behavior identified in Rasmussen's (1983, Rasmussen & Jensen, 1974) skill-rule-knowledge hierarchy. The skill-rule-knowledge hierarchy categorizes human error according to three levels of cognition at which the errors occur, as well as for deconstructing task environments into components that fall into each level as a way to grasp *a priori* factors that might induce human error (Reason, 1990). The skill-based level of human performance encompasses very low-level cognition such as stored patterns of stimulus representations and physical motions. Errors at this level mainly stem from variability of physical force, space, and temporal coordination. The rule-based level consists of IF-THEN rules of actions to take when learned, pre-defined conditions are met. Errors at the rule-based level include the misclassification of situations leading to the application of the wrong rule or an incorrect recall of the action component of a rule. Finally the knowledge-based level of cognition refers to the conscious, attention-directed

processes that often rely upon stored knowledge or the effortful transformation of knowledge. Actions at this level must be planned on-line, using conscious analytical processes. Errors at this level arise from resource limitations ("bounded rationality") and incomplete or incorrect knowledge.

James Reason greatly expanded upon the skill-rule-knowledge hierarchy in his 1990 book, *Human Error*. Reason begins with the notion that traits that enable fast and flexible human cognition in most environments can become detrimental when misapplied. Reason coined the term "cognitive balance sheet" as a metaphor for human cognitive strengths being in some situations weaknesses. All systematic human error is a result of the misapplication of some cognitive process at one of the three levels of the skill-rule-knowledge hierarchy. The errors modeled by the CSM fall within the realm of lapses as defined by Reason (1990), and the list below describes all the different error types that the CSM accounts for.

1.1.1.1. Capture Errors

Capture errors can occur in CSM's output when an environmental source of activation becomes too strong relative to the top-down activation the schema receives. Capture errors can also occur when schema competition is inappropriately resolved, such as when self-activation in the schema network is very high.

1.1.1.2. Omission and Anticipatory Errors

Omission errors result when a schema simply is not activated past threshold, and hence not selected. Schemas also might fail to be selected, or their execution prohibited, by the inability to select appropriate object arguments or resources for the corresponding actions. The former scenario may result in an entire subtask failing to be executed. The

latter scenario may result in an anticipation error, wherein an action that should have been performed later in a sequence is attempted before its preconditions have been satisfied.

1.1.1.3. Perseverative Errors

Perseverative errors may arise if self activation is too great or if lateral inhibition is insufficient. In these cases schemas fail to be deselected at the appropriate time. Apparent perseveration may also arise in the case of a perseverative object substitution, such as if the representation of an object remains active even after that object's use.

1.1.1.4. Object Substitution Errors

Object, as well as place, substitution errors may arise when the schema's correct arguments do not have the most active representations when the schema is selected. This may occur when, for example, noise is high and/or the excitation of object representations by schemas is insufficient.

In summary, CSM proposes that different error types result from the actions of different representational structures. These structures each produce errant behavior due to the interactions of activation influences from task, environment, and top-down sources.

1.2. A Recurrent Connectionist Approach to Selection of Routine Sequential Action

Botvinick and Plaut (2004) take a radically different approach in their theory of human action selection in routine sequences. Rather than proposing a symbolic interactive activation network, Botvinick and Plaut argued for a parallel distributed processing (PDP) approach to action sequence representation and selection. PDP approaches are characterized by the distributed representation of information which emerges out of the entire network of simple processes. This is in contrast to CSM's approach with its discrete, isolable structures that embody representations. Also

characteristic of PDP, detailed mechanisms must develop through learning, and because of that they are tied to the structure of their task domains. In particular, Botvinick and Plaut take issue with hierarchical models of human cognition, like Cooper and Shallice's (2000) CSM, saying it is unlikely that the brain is structured such that it mirrors its environment. Instead, they advocate a general learning mechanism capable of learning from samples of its environment and representing what it needs to know in networks of simple structures, with behavior emerging from the interaction of network constituents. Schemas and goals, they argue, are epiphenomenal.

The basic principle underlying PDP models is that the activation of each node is based on excitation and inhibition received from nodes linked to it through weighted connections (Botvinick and Plaut, 2004). Often the nodes are segregated into three layers, with a first layer carrying a pattern of activation representing some input to the system. Activation propagates from this input layer through an internal or hidden layer, which transforms the input, sending a pattern of activation to an output layer whose nodes together represent the system's response to the input. A network is "recurrent" when loops or circuits can be traced through its set of connections. Recurrency gives the network a representation which reflects task demands in the context of prior internal states (Elman, 1990). In the simple recurrent network (SRN), every hidden node is connected to all nodes in both the input and output layers. Additionally, in Botvinick and Plaut's SRN every unit in the hidden layer was connected to every other unit in the hidden layer (see the diagram depicted in Figure 3). A critical aspect of recurrent connectivity is that it allows information to be preserved and transformed across time.

Each step of processing carries information about the state of the system at the previous time step, thus the system is sensitive to temporal context.

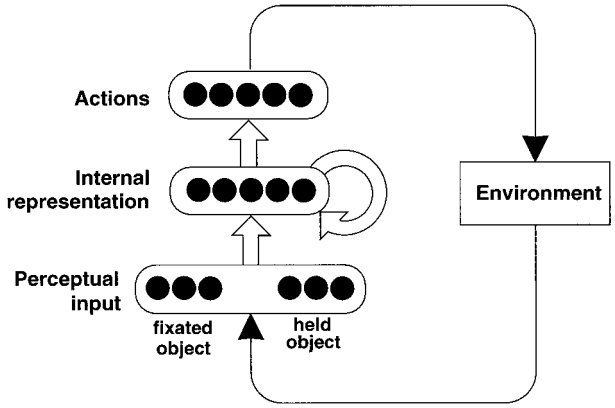

Figure 3. Architecture of the overall SRN model from Botvinick & Plaut (2004). Open arrows indicate that every unit in the sending layer is connected to every unit in the receiving layer.

Botvinick and Plaut (2004) trace their assumptions concerning task structure back to Lashley (1951). In the early 1950's, researchers often viewed sequential behavior and tasks as having a strictly linear structure (Botvinick & Plaut, 2004). Lashley rejected this notion and claimed that tasks and behaviors that perform those tasks usually have some degree of hierarchical structure, since identical and nearly-identical behaviors are often performed in different contexts to perform the same or nearly-same tasks. Any cognitive representations of the tasks might not need to assume a hierarchical structure themselves, but they must be able to account for task context. A model of human action selection, therefore, does not necessarily need to mirror a task's structure in its own structure – the two structures are separate. Indeed, Botvinick and Plaut reject the notion that a cognitive representation's structure must mirror the structure of the thing it represents.

Botvinick and Plaut echo Lashley's sentiment that some actions appear in multiple contexts (e.g., stirring sugar or honey into tea, stirring sugar or cream into coffee: stirring is essentially the same). The problem with associationist accounts is that there is no

accounting for context and the selection of one action in multiple contexts. Clearly many tasks have a hierarchical or quasi-hierarchical structure that capture some measure of task context – the appearance of one subtask in many supertasks. The problem then becomes one of how to select a task appropriate to some representation of a context. However, Botvinick and Plaut are suspicious of models such as Cooper and Shallice's wherein mental representations mirror task structures. They argued that task structure and context should be captured in a distributed representation by a generalized learning mechanism.

Botvinick and Plaut (2004) applied their SRN model to the same coffee preparation task used by Cooper and Shallice (2000). In Botvinick and Plaut's (2004) model, each node in the input layer corresponded to each object present on the breakfast tray. Additionally, they experimented with having their model prepare tea as well as coffee, and so the model had two additional nodes: one to represent an instruction to prepare coffee, the other, an instruction to prepare tea. Tea preparation was added to demonstrate a theory which could account for tasks which have overlapping steps, such as "pour water" and "add sugar."

Training the model involved running it on a trial, then propagating an error signal backward through the network in order to adjust inter-node connection weights. After many thousands of cycles of attempting the task and adjusting weights, the model could produce both coffee and tea preparation sequences, including two alternates of both tasks in which sugar was obtained either from packets or from a bowl.

At test, the SRN model was given a simulated environment, and in some cases also an activation of an instruction node (either "prepare coffee" or "prepare tea"). Activation propagated from input nodes to hidden nodes, which in turn would transform the input

and pass on a pattern of activation to the output. The simulated environment changed according to the SRN's output, which in turn closed the model's perception-action loop by triggering a new pattern of activation in the model's input layer. The perception-action cycle continued until the task was complete, at which time the model took no further action.

1.2.1. Modeling Human Error in a Simple Recurrent Connectionist Network

To the SRN, a subtask is distinguished solely statistically, that is by the sample procedures the SRN is exposed to during training. Subtasks are determined by local associations and by branch points. The local associations come from each of its steps always being associated with one particular next step. The branch point is the end of the subtask where the next step performed could be from one of several different subtasks. Botvinick and Plaut's (2004) SRN account of routine action selection was driven by the statistical nature of its learning algorithm and the network's holistic representation of the task context. Steps that always appeared together in a particular order became represented as one subtask, while whole subtasks might vary in the order of their appearance in training. The SRN's context representation was composed of the task's main goal and the steps accomplished. The SRN used distortion of propagated activation of the context representation to account for human error. Specifically, the SRN posited that distortion occurring in the middle of a subgoal leads to capture errors at the end of that subgoal when the context representation begins to resemble another task's context.

Botvinick and Plaut (2004) assume that all lapses result from a degradation of representation of task context, "An error occurs when the network is in some situation calling for some action and distortion causes its context representation to resemble a

pattern the model has learned to associate with a different situation and a different action." (p. 409) Therefore their strategy for modeling error in routine tasks is to introduce noise into the activation pattern of the hidden layer during performance of the coffee/tea task. They found that lapses such as capture errors and omissions tended to occur when people were supposed to perform some action that came at the boundaries of subtasks within the hierarchical structure of the task (as opposed to the structure of the task's representation within the model). For example, in the coffee making task capture errors were more common on the first or last step of the sugar subtask (e.g., "grasp spoon" or "open container") than on some step in the middle of the sugar subtask (e.g., "scoop sugar" or "stir"). Furthermore, capture errors were more likely to occur if the capturing task had been performed more times than the captured task. The errors resulted from degradation of the model's representation of task context, and the model tended to respond on the basis of a representation's similarity to a more familiar action.

1.2.1.1. Capture Errors

For example, in the coffee making task capture errors were more common on the first or last step of the sugar subtask (e.g., "grasp spoon" or "open container") than on some step in the middle of the sugar subtask (e.g., "scoop sugar" or "stir"). Furthermore, capture errors were more likely to occur if the capturing task had been performed more times than the captured task. The errors resulted from degradation of the model's representation of task context, and the model tended to respond on the basis of a representation's similarity to a more familiar action.

In particular, Botvinick and Plaut (2004) found that their model was susceptible to context representation disruption, and then capture error, when noise was introduced in

the middle of a subtask. When context confusions occur, the effects tend to be felt at branch points (subtask boundaries), or junctures at which the immediately preceding actions and/or environmental context bear associations with subsequent actions other than the correct sequence for that task. An error in performance occurs at the branch point even though a drift in contextual representation may have begun several steps earlier, toward the middle of a subtask. In fact, the SRN model's account of context representational drift predicts that capture errors are more likely to result when distraction occurs in the middle of a subtask, rather than immediately before a branch point. Apparently context representation is particularly vulnerable to distortion near the middle of a subtask, because the SRN model's pre- and post-step contextual representations become more similar in the middle of a subtask compared to at the beginning or end of the subtask. Indeed, their model's account predicted empirical findings in a later study by Botvinick and Bylsma (2005).

1.2.1.2. Omission and Anticipatory Errors

Omission and anticipatory errors resulted from exactly the same mechanism that caused capture errors. It just so happened that the distorted contextual representation resembled the context for a sequence from later in the same task, rather than from another task as in capture errors (e.g., leaving out the sugar subtask and skipping directly to cream adding and then to drinking).

1.2.1.3. Perseverative Errors

Similarly, perseverative errors occurred when the inserted action sequence happened to come from earlier within the task being performed.

1.2.1.4. Object Substitution Errors

Curiously, the SRN model exhibited no object substitution errors when subjected to low levels of noise thought to mimic distraction in neuro-intact individuals, but did produce a small number of object substitution errors when subjected to high levels of noise in a simulation of neurologically impaired patients with action disorganization syndrome.

1.3. Recent Developments in the Debate Over Schema versus PDP Representation of Routine Action Selection

Psychological Review published an article by Cooper and Shallice (2006a) in reply to Botvinick and Plaut's (2004) account of action selection. Cooper and Shallice contrast the PDP account with their own CSM account, concluding that abstract, symbolic representations of causal elements of behavior, namely goals and schemas, are necessary to reproduce the range of flexible behavior seen in humans. Cooper and Shallice also criticize the SRN theory of action selection for being too dependent upon its training context to be able to generalize a routine task to a new environment or interface or to be able to interchange action subsequences as humans often do. Cooper and Shallice also criticize the SRN model for saying nothing about the role of distractor objects in action selection or about how object substitution errors may occur in neuro-intact individuals. Botvinick and Plaut (2006), in the same issue, claim that Cooper and Shallice (2006) mistook several superficial implementational issues for fundamental theoretical positions, underestimated the computational power of recurrent networks as a class, and in some ways mischaracterized the relationship between the SRN and CSM accounts.

The debate published in that issue of *Psychological Review* concluded with a brief retort by each side. Cooper and Shallice (2006b) raise concerns with several of the

implementational adjustments of Botvinick and Plaut (2006) and criticize the other authors for not examining potential interactions of their adjustments. Furthermore, Cooper and Shallice maintain their position that the SRN model is not a sufficiently abstract account of routine action selection, that SRN does not produce any action subsequence which it does not encounter in its training environment, which is problematic, they argued, because humans do produce novel subsequences. Cooper and Shallice maintain that goals are essential to many human behaviors, particularly because goals typically direct routine actions. And they criticize the SRN account for saying little about the relationship between routine and non-routine action selection systems.

Botvinick and Plaut (2006) reply that the link between errant behavior and prior experience is a strength of their model, rather than a weakness. For the SRN model, the training set is of paramount importance because the SRN model is a kind of statistical sampling and acting machine. What can vary in the test set varies in the training set, and the training set is a representative sample of the test set. So learning any specific task takes place within the context of learning a broad variety of other tasks. Botvinick and Plaut assert this kind of learning by broad statistical sampling is an important strength of their theory because it possesses strong ecological validity. Finally they concluded that moving forward will likely involve building a theory of non-routine action selection and connecting it with a theory of routine action selection. But, they said, there is frustratingly little empirical account of how a non-routine action selection system might work alone or in concert with a routine action selection system in humans.

1.4. Other Theories of Action Selection

GOMS (Card, Moran, and Newell, 1983; John, 2003) and production systems such as ACT-R (Anderson et al., 2004) incorporate implicit theories of human action control for routine procedures. GOMS, Goals Operators Methods and Selection rules, is a framework for analyzing tasks humans engage in and making quantitative predictions about performance given a particular interface with which to accomplish that task. GOMS is based on a stage model of human information processing, and as such it is dependent upon a psychological framework. The fundamental assumption GOMS makes is that both the task structure and the cognitive architecture are necessary to describe and predict human performance.

GOMS was in particular designed to model and predict skilled human performance of routine tasks, and all of its predictions carry the assumptions that the behavior being modeled is both skilled and routine. Therefore, like Cooper and Shallice's contention scheduling model, all behavior is conceived as being completely goal-directed. Methods (actions) are well-learned sequences of subgoals and operators (low-level actions) that can accomplish a goal (John, 2003). When more than one method is possible, selection rules specify under what circumstances which method is selected. It is assumed that, because GOMS models skilled behavior, the most efficient method will be selected.

As GOMS was conceived as an engineering tool for predicting completion times and suggesting designs for task and interface structures in the skilled performance of tasks, it generally has little to say about human error. This is rooted in the fact that humans generally commit few errors when engaged in a well-practiced routine behavior, and so Card, Moran, and Newell (1983) approximated skilled, nearly-error-free human behavior in their engineering model by simply not being concerned by it. But this is not to say that

GOMS cannot be applied to the study of erroneous action selection. In fact, a recent study by Wood and Kieras (2002) applied GOMS to a system redesign task and found that it could function as a tool to predict human error and mitigate factors contributing to it in the design of system interfaces.

Wood and Kieras' (2002) approach assumes a modified version of the five distinct error stages described by Card, Moran and Newell (1983):

1) Error. The user makes a mistake.
2) Detection. The user is aware an error has occurred.
3) Identification. The user identifies the error's type.
4) Correction. The user corrects the effects of the error.
5) Resumption. The user resumes normal tasks.

Wood and Kieras' (2002) general framework for error recovery specifies the infrastructure needed to model erroneous behavior (Figure 4). Once a GOMS model has been constructed, its static and dynamic aspects may be examined by the modeler to identify sources of error. These sources can be procedural or non-procedural, where procedural aspects stem from the effects of the action sequences in the methods and non-procedural aspects arise from perceptual-motor factors. In particular, Wood (2000) described several error types that may be identified by GOMSL (Kieras, 1999) analysis, how patterns in the GOMSL analysis identify the potential error, and suggests remedial design guidelines.

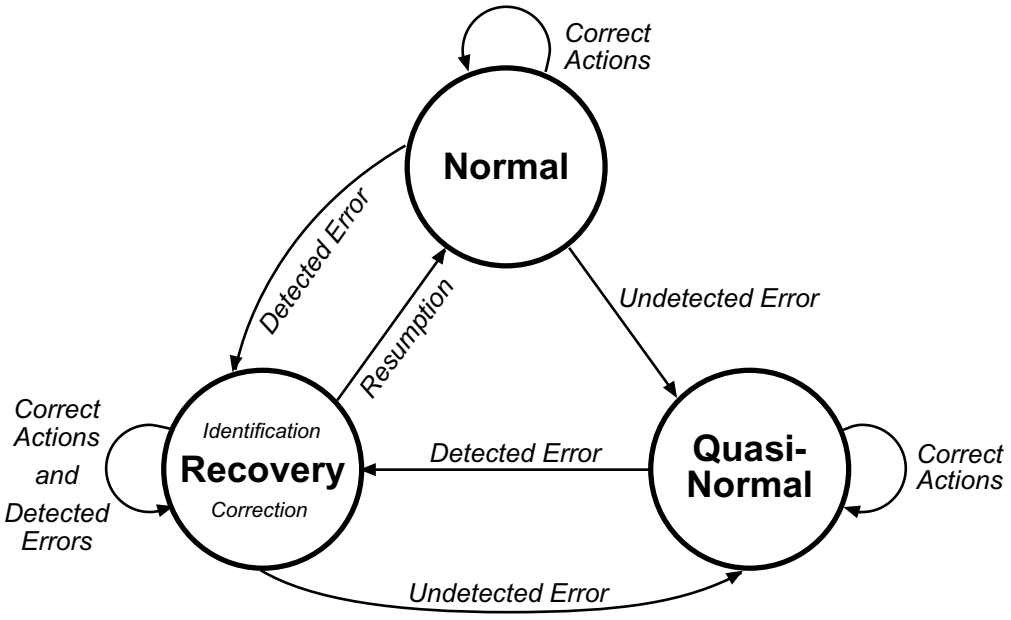

Figure 4. Wood and Kieras' (2002) general framework for error recovery. The state diagram illustrates user mental states while moving through error recovery stages.

For example, after describing what a capture error is, Wood describes the GOMSL pattern of output for that error and design guidelines for remediating interface factors that may contribute to inducing capture error in the user's performance:

> *Pattern.* The general pattern includes a sequence of steps that are used to accomplish multiple task goals, followed by a Decide (the divergence point), and ending with different action sequences. Capture errors are most likely when the action sequences are dominated by well-practiced motor actions.
>
> *GOMSL Example*.
>
> ```
> Method_for_goal: Save file
> Step 1. Keystroke ":".
> Step 2. Keystroke "w".
> ```

```
Step 3. Decide: If "finished writing" then
        Keystroke "q".
Step 4. Return-with-goal-accomplished.
```

This classic example (Norman, 1983) from the UNIX vi text editor shows a method for saving a file that includes the option of quitting after saving. To save the document, the user must type the sequence ":w". To save and quit, the user must type ":wq". If the save-and- quit sequence is used much more than the save sequence, the user will have a tendency to save-and-quit when the intention is to save and continue.

Design Guidelines. Norman (1983) and Lewis and Norman (1986) suggest three possible guidelines to minimize capture errors. First, minimize overlapping action sequences when possible. Second, if that is not possible, put in a verification check at the divergence point. In the above example the system could force the user to confirm quitting. The third guideline is to provide adequate system state information to the user. Although not relevant to the example given here, system state information can sometimes cue the user that the system is in a different state than expected.

Wood's and Kieras' (2000; Wood & Kieras, 2002) effort forms the beginning of an engineering approach to identifying and mitigating human error by addressing error-inducing elements of interfaces and tasks early in the design process. While such work surely possesses much practical value, its level of analysis is not meant to discover the cognitive mechanisms responsible for human action sequence representations and how they produce correct and errant behavior.

ACT-R, however, is a cognitive architecture intended to be a generalized theory of human cognition (Anderson et al., 2004). As such it must be concerned with those mechanisms of human action selection and their resultant correct and errant behaviors. ACT-R is a hybrid production system which uses symbols to transmit information between distal modules, each responsible for different types of computation (e.g., visual perception, motor movements). The modules themselves use subsymbolic processes to govern their computations. Production systems use a collection of IF-THEN rules to specify actions to be taken and their preconditions. In ACT-R, these preconditions can come from any of a number of sources, including sensory information and goals. Like contention scheduling, production systems in general and ACT-R in particular govern their action selection based on discrete, symbolic knowledge structures. But in ACT-R's case, some governance, such as conflict resolution when multiple actions match a set of preconditions, relies on processes dependent upon continuous, subsymbolic processes which often act upon the symbolic structures.

Lebiere, Anderson, and Reder (1994) demonstrated that human-like errors can be successfully generated by a production system, in their case, a model built on the ACT-R cognitive architecture. Errors modeled were from an algebra task wherein subjects had to memorize a digit span of 2, 4, or 6 digits and then solve a linear equation before recalling the digits. Equations were simple or complex, with one or two transformations required to solve. Human errors increased both with increasing equation complexity and increasing digit span, the factors being additive. Most errors occurred in algebraic transformations (such as in forgetting to invert a sign), though many others were arithmetic errors where subjects retrieved the wrong fact in the addition or multiplication

table. The authors replicated errors of omission (such as forgetting to invert signs) by applying a latency threshold for the retrieval of declarative memory facts. Retrievals failed if a fact did not receive enough activation from its source context. Errors of commission (like arithmetic errors) resulted from allowing for imperfect retrieval from long term memory. The wrong fact was retrieved if it partially-matched the context and the stochastic nature of its activation computation happened to give it a higher activation than the correct fact.

Byrne (2003) highlighted other mechanisms in ACT-R that could be used to model human error, particularly for predicting human error in routine procedures. One area likely to yield errors, even when the "correct" knowledge is known by the operator, is the procedural memory mechanism. ACT-R chooses productions based on their utility, but the process is noisy. Thus when there are multiple viable alternatives, ACT-R will choose stochastically from among them, possibly choosing an action that fails to achieve the current goal. Likewise, ACT-R's declarative system provides rich ground for potential error generation as Lebiere, Anderson, and Reder (1994) described.

Chung and Byrne (2008) implemented Byrne's (2003) framework in a computational model of a routine procedure. The procedure had been designed to induce a particular type of error, *postcompletion error* (Byrne & Bovair, 1997) wherein operators neglected to take a step required after the main goal of the task has been accomplished. Chung and Byrne used a declarative memory mechanism based on Lebiere, Anderson, and Reder's (1994) retrieval latency threshold to mimic human operators' postcompletion error rate. Chung and Byrne's human data showed that postcompletion error is eliminated if a highly-salient and sufficiently specific cue is used to aid retrieval of memory for the

postcompletion step. They were able to show that the cue works by making it immediately accessible to the model's simulated visual system. The cue appeared just-in-time with the postcompletion step. The model had a production specifying that if the cue object were to appear, then perform the postcompletion step.

1.5. Recent Empirical Findings

Cooper and Shallice (2006a, 2006b) alluded to the possibility that objects in the environment could have important consequences for action selection, since often our actions are performed on some object or using some object as a tool. Objects all exist in space, and how they are arranged in space can impact human performance of routine tasks, both in terms of speed and error commission.

Recent data from our laboratory indicated one factor particularly important to performance of routine procedures, visual layout. Chung (2006) found that users relied much more on the spatial layout of interface control elements (checkboxes, radio buttons, etc.) to guide their action selection than they relied upon control element labels or goal structure. Chung's study employed two experiments each using two quasi-isomorphic procedures performed with software interfaces, Phaser and Transporter. Participants trained on both tasks and then tested one week later. Experiment 1 changed both interfaces halfway through testing, with the Phaser's labels each being changed to a row of X's (Figure 5).

The Transporter interface was laid out so that controls were grouped into clusters by subtask (Figure 6). Controls in the Transporter's first subtask's cluster were mixed together so that there was no clear relationship between their spatial arrangement and the order in which they were used. Initially the second subtask's cluster was arranged so that

the first control to be used in that subtask was the bottom-most element in the cluster, with subsequent actions using the next-lower control. The cluster for the third subtask was laid out top-to-bottom. After the interface change, the positions of controls within each of these clusters reversed so that the second cluster became top-to-bottom and the third cluster bottom-to-top. The first cluster reversed the order of its controls, going from top to bottom.

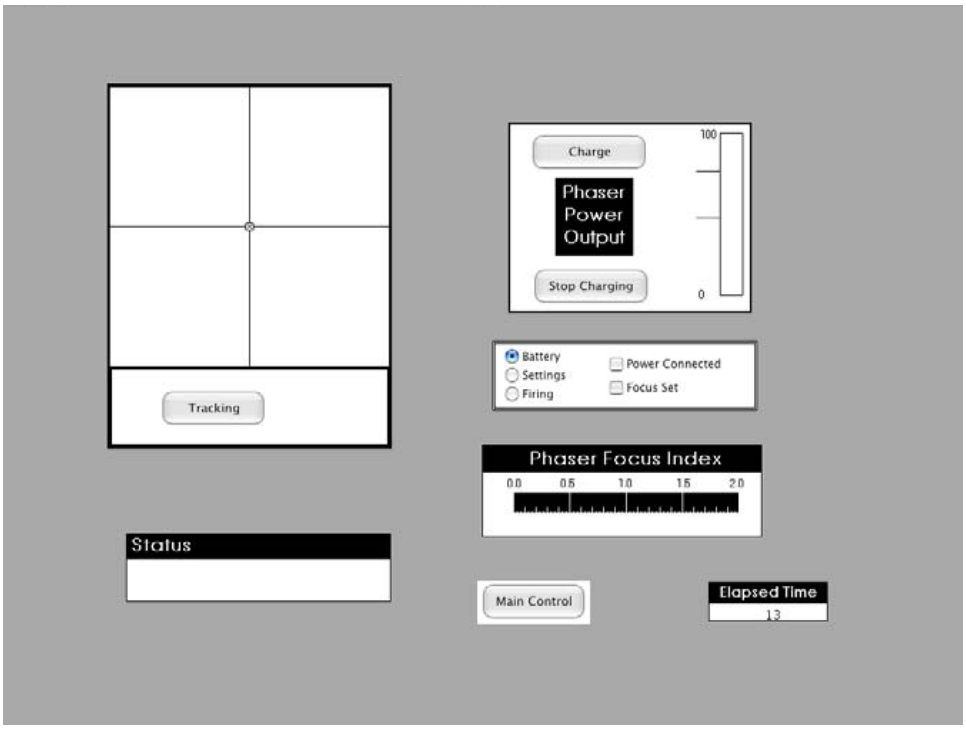

Figure 5a. Label removal: Pre-change Phaser interface.

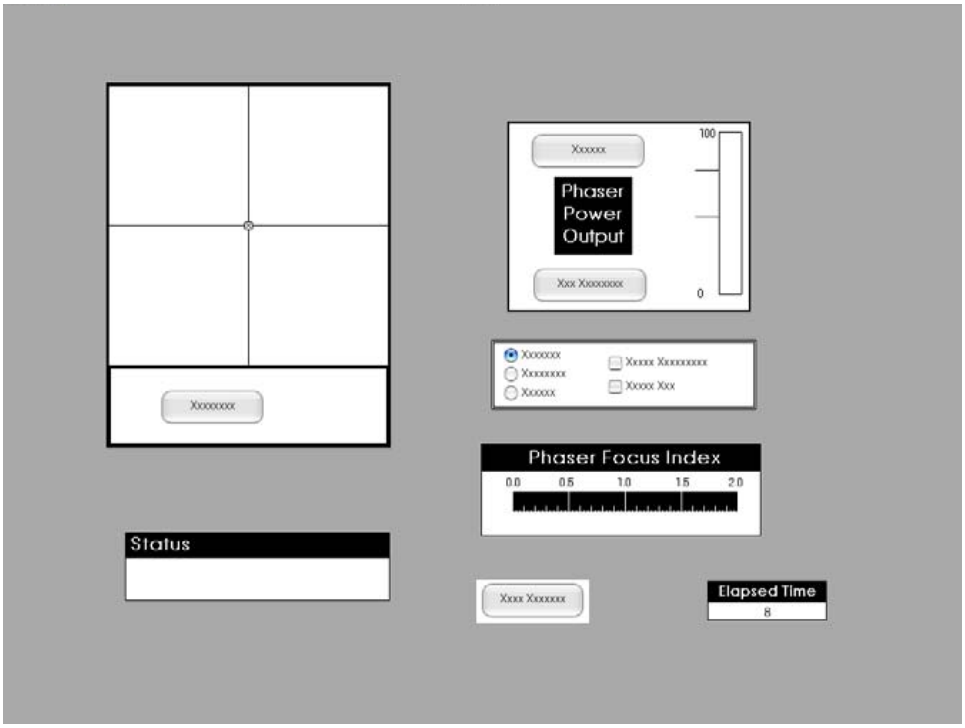

Figure 5b. Label removal: Post-change Phaser interface.

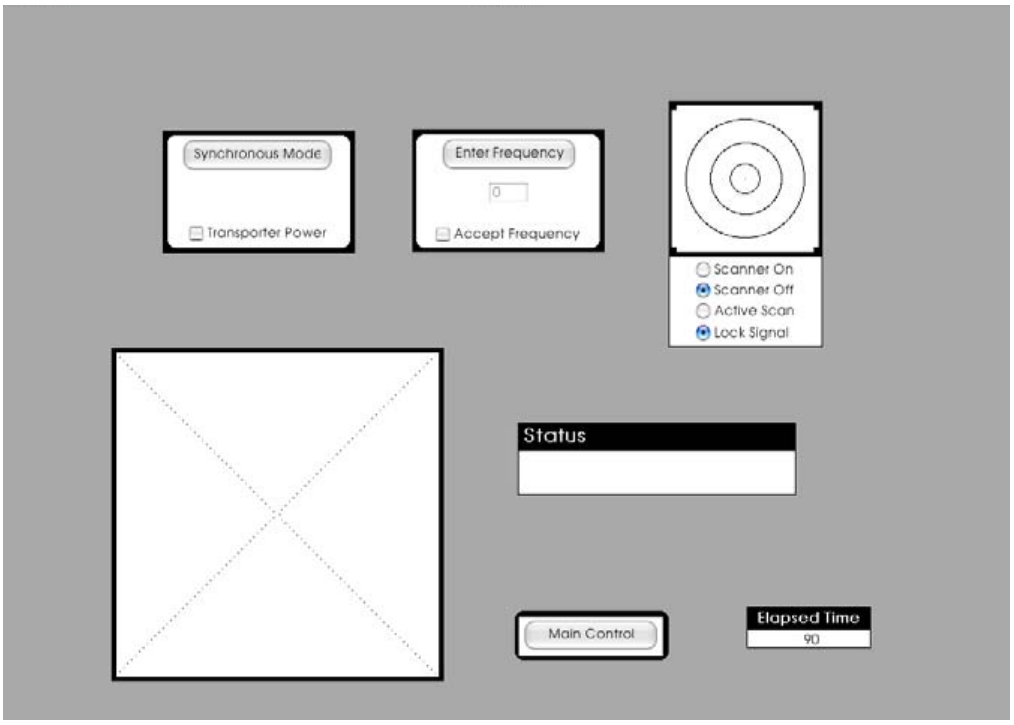

Figure 6a. Layout change: Pre-change Transporter interface.

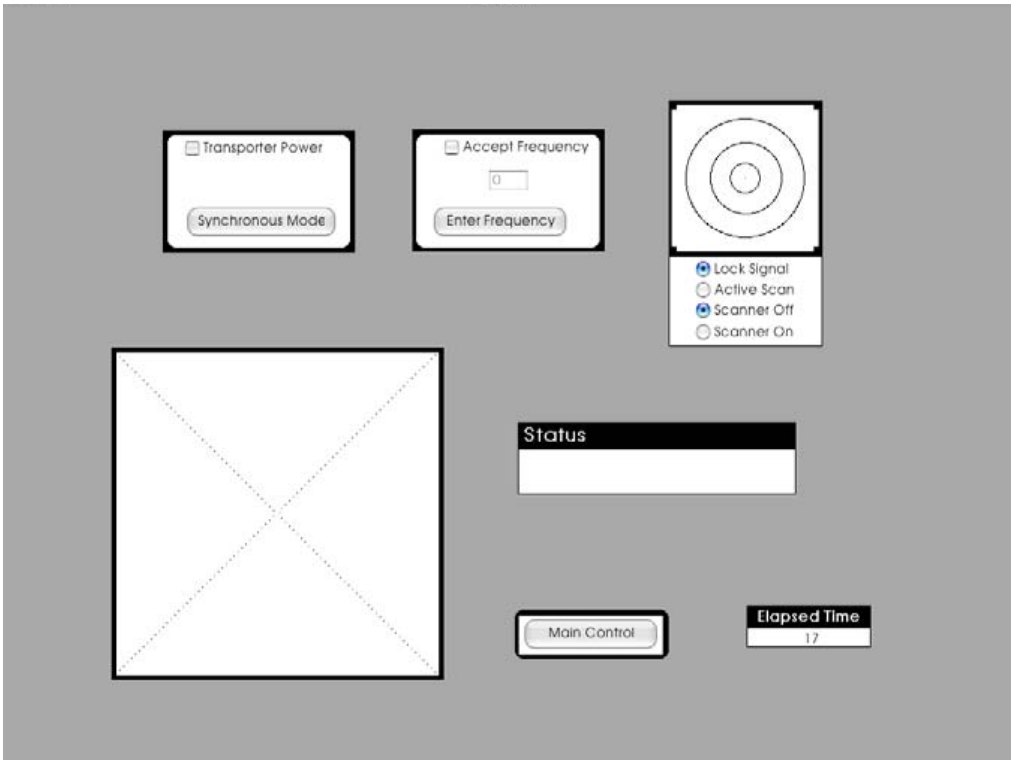

Figure 6b. Layout change: Post-change Transporter interface.

Chung found that removing control labels in the Phaser resulted in only a slight and non-reliable increase for error rate on the first step of the task, while there was no change in error rate for other steps. The pattern was the same for Phaser task step response times. As for the Transporter, error rate did increase with the change in layout, particularly on the first step. There were no effects of layout change on Transporter response times, however.

In Experiment 2, the Transporter changed from its down-up layout to one in which all three clusters had controls arranged in a top-to-bottom order. Chung found that error frequencies were lower for several steps for the post-change Transporter than for the pre-change Transporter. For the Phaser, since controls were not arranged with all of one subtask's controls in the same cluster, color-coding was added such that all controls of the same subtask were highlighted in the same color. If subjects used task structure to guide their action selection, then providing visual interface cues regarding subtask groupings should have aided performance. Instead, Chung found that error frequencies drastically increased after the Phaser interface change, despite subject reports stating that they used color coding and felt as though their performance improved because of it. Chung speculated that globally useful rules specifying the next location at which an action is to be performed can help navigation in interfaces used for routine procedures because cognition is conserved relative to interfaces that do not lend themselves to such rules. Other work (e.g., Fu & Gray, 2004; Gray & Boehm-Davis, 2000; Gray et al., 2006) has demonstrated the humans tend to behave in locally-efficient manners that conserve cognitive and perceptual-motor resources during interactive behavior. If humans are such "cognitive misers," as Gray asserts, then it is likely that people would draw upon global

navigation rules during interactive behavior and integrate those rules into their skilled routines.

Other recent but as yet unpublished data from a later experiment employing the Phaser and Transporter tasks continued to examine the role spatial layout may have in human performance of routine procedures. Table 1 lists the tasks, manipulations, and major findings from this latter experiment. From the empirical data reported by Chung (2006) and from the subsequent experiment, it seems clear that spatial layout of an interface used for a routine task can have profound effects on human performance of that routine task.

Table 1
Tasks, Manipulations, and Major Findings from Unpublished Star Trek Procedure Experiment

Task	Manipulation	Major Findings
Phaser	move clusters	subjects slower on some steps by 200 – 500 ms per step
Phaser	move clusters & move buttons within clusters	subjects generally slower by 500 – 1,000 ms per step
Transporter	add extraneous buttons	subjects generally slower by 500 ms per step
Transporter	move clusters, move buttons within clusters, & add extraneous buttons	subjects' error rate generally jumps from 0% – 5% to 15% – 20% and slower by 1000 – 2000 ms, per step

GOMS is not likely to provide an explanation for how people really go about selecting routine actions because it was simply not designed for the task of advancing basic cognitive science theory, though as an engineering model GOMS will certainly benefit from such advancements. CSM may be in a better position to advance to more ranges of behavior because of its more abstract level of specifications, though at low-

levels of computation SRN may very well provide explanation. ACT-R perhaps is in the best position to integrate action selection in routine and non-routine behaviors because it is a generalized theory of human cognition.

As an example of the importance of using a generalized theory of human cognition, bear in mind one of Cooper and Shallice's (2006a, 2006b) criticisms of the SRN model, that objects not fixated or grasped during training effectively do not exist. ACT-R has enough of a simulated visual system to be potentially influenced by distractor objects in the environment. It is difficult to imagine how the SRN model would mimic human 500 ms increased response time as an effect of having extraneous buttons added to the Transporter. The layout change data from our lab clearly indicate a strong influence of visuo-spatial features of the interface on action selection human performance and error and therefore seems like a fruitful space for exploration. Other recent work (Gray & Fu, 2004; Gray et al., 2006) indicates that people are flexible in how they accomplish their tasks at small time scales, typically five seconds or less, and use combinations of their own memory and the perceptual-motor properties of their environment to achieve close to optimal efficiency in their performance at that time scale. If that is true and visuo-spatial factors can influence action selection, as the CSM account indicates, then people may use their environment as cues for the selection of actions. Indeed, CSM actually includes environmental triggers, or cues, as one source of potential schema activation. Furthermore, Chung and Byrne (2008) found that cuing by the interface mitigated postcompletion errors. It therefore seems evident that perceptual-motor factors do play an important, interactive, role in action selection.

How are actions and action sequences represented? The CSM and SRN accounts both offer views that are impressive in the range and depth of behavior they explain. SRN offers an account of generalized statistical learning of action sequences that is compelling, but CSM, being more abstracted, seems as though it would explain a larger range of behaviors at the rule-level of human behavior and is possibly in a better position to interface with other cognitive systems that would handle the knowledge- and skill-levels. The ultimate test of our grasp of action selection and errors thereof, at all three levels of the skill-rule-knowledge hierarchy, is likely to come from the practical value of engineering tools designed to assess those issues. GOMS is making promising progress in that respect, and feedback regarding the predictive efficacy of tools like GOMS will give us useful information about how well we know these phenomena.

How do environmental factors like spatial layout and task structure influence action selection? It is clear from work like Chung (2006) and Chung and Byrne (2008) that spatial layout does impact human performance of routine procedures. It may be that in learned action sequences, the representations of where those actions occur are tightly coupled with the representations of the actions themselves, as evidenced by Chung's findings from the Transporter task.

What can human error tell us about how actions and action sequences are represented, and how spatial layout and task structure influence action selection? The fact that subjects in Chung's (2006) Transporter clicked the wrong control object with no accompanying increase in response time, rather than taking more time to make certain the right control object was found, indicates a lack of speed-accuracy tradeoff. Instead, subjects appeared to have a high willingness to act based on sequential action-location information alone.

Once a corpus of human error-related factors such as spatial layout change has been built, design guidelines may be established that can inform the design of interfaces that have a low incidence of inducing operator error.

It is clear that more empirical work needs to be done to map the spatial and task structure factors that influence human action selection, and how those factors contribute to selection of the wrong action. Clearly a foundation built in a generalized theory of human cognition is needed in order bind these factors and their interactions together. ACT-R has already showed promise as a tool for investigating human action sequence representations and error in routine action. Further study should use a combination of extensive empirically-acquired behavioral data and cognitive modeling to map these factors and tie them together into a cohesive account of human action selection.

2. BEHAVIORAL STUDIES

I performed two experimental studies and two modeling studies aimed at obtaining the kind of empirical behavioral data and insight from modeling to begin to tie perceptual-motor and cognitive factors into a cohesive account of human action selection. General theories of cognition, such as ACT-R, come with mechanisms for generating cognitive, perceptual, and motor predictions that are all based upon empirical research. Many general theories of human cognition, like ACT-R, have theories of perceptual and motor processes for good reason: human cognition is situated within those systems and uses those systems to exchange information with the world. Mating a framework like ACT-R with a successful theory of action selection will provide us with a richer view of routine procedural behavior than what a theory of routine procedural behavior alone could provide us.

If we are to move toward that cohesive account of human action selection, we should test whether CSM and SRN are basically correct in their accounts of sequential human task performance in the presence of the perceptual-motor factors discussed by the likes of Chung (2006) and Chung and Byrne (2008). To that end, the two experiments I performed used a version of Byrne and Bovair's interactive Star Trek-themed procedural task. Byrne and Bovair's task, being formulated specifically for research, were novel procedures for the subjects. The procedures were set in the fictional world of Star Trek to encourage engagement of the undergraduate participants (Byrne, Maurier, Fick, & Chung, 2004). The experiments manipulated the structures of the two tasks tasks in order to test the CSM and SRN accounts of human task representation.

The first experiment tested the SRN's assumption that subtasks are delineated strictly on the basis of step co-occurrences. Experiment 1, using Byrne and Bovair's Phaser task, presented all subtasks in the same order on every trial, but segregated them by perceptual means such as grouping and by semantic means such as object label similarity. If, however, representations of task structure are entirely based on step associations and a holistic representation of task context as the SRN claims, then manipulating step order within a subtask should completely destroy humans' ability to perform that subtask. Thus Experiment 2, using Byrne and Bovair's Transporter and a new task, the Jammer. Experiment 2 delineated subtasks by co-occurrence as the SRN assumes, but it also reordered the steps within one subtask in two conditions. In one other condition, it manipulated the order of subtasks within the procedure.

Experiment 2 ran concurrently with Experiment 1, using the same subjects. To the subjects, the Experiment 2 tasks – the Transporter and the Jammer – appeared as one

experiment together with the tasks of Experiment 1. Thus, design, procedures, materials, and participants for Experiment 2 were identical to those for Experiment 1 except as noted.

2.1. Experiment 1

2.1.1. Introduction

Experiment 1 used the Star Trek Phaser procedure (Byrne & Bovair, 1997; Chung, 2006; Chung & Byrne, 2008) to manipulate subtask delineation. Experiment 1's Phaser procedure put one, partially-completed subgoal on hold while the subject started and completed one other subtask. Then the subject returned to the first subtask to complete it. Subtask coherence was established by placing all controls for each subtask within their own cluster within one bounded area of the interface and using semantically-similar step names as measured by Latent Semantic Analysis.

The intervening subtask used step names that were all similar to each other, but dissimilar to the paused subtask. One group of subjects trained on this procedure from the outset. Another group trained on a procedure that kept all subgoals intact, then after completing one-half of the trials during the testing session, the procedure changed to the intervening-subtask procedure. Still a third, control group received the intact condition throughout training and testing. A fourth group of subjects worked with a semantic control version of the basic Phaser in which no control label was similar to any other control label. Furthermore, the Phaser task did not adhere to the SRN's working definition of a subtask: subtask order never varied, and subtasks were delineated only by spatial arrangement of control objects and semantic relatedness of object labels.

2.1.2. Method

2.1.2.1. Participants

Ninety-two Rice University undergraduates participated in Experiment 1 to earn either course credit or $25.00. All subjects who finished the experiment were eligible for the cash prize competition. Five subjects dropped out of the study or were removed due to technical error with the experiment software. The remaining 87 subjects had a mean age of 19.9 (1.5) years and 48 of them were male, 39 were female.

2.1.2.2. Design

Experiment 1 used four between-subjects conditions for the Phaser task. The Phaser conditions were basic Phaser (no intervening subtask), trained subtask pausing, untrained subtask pausing, and a semantic control version of the basic Phaser.

2.1.2.3. Materials

As in previous studies using the Star Trek paradigm, Experiment 1 used a set manuals of five or six pages to instruct subjects on the experiment's procedures. The manuals featured an overview of the procedure, detailed instructions complete with example figures, and a chart illustrating the task's procedure. Copies of all manuals are available for download from this dissertation project's website, http://chil.rice.edu/tambo/dissertation/ .

Experiment 1 was programmed in Lisp and run in the Macintosh Common Lisp (MCL) environment version 5.1 on eMac Macintosh computers running Macintosh OS X 10.2 and 10.3 and each equipped with a standard Apple single-button mouse, standard Apple QWERTY keyboard, and Sony MDR-201 headphones. Subjects also participated in a web-based post-experiment survey displayed in Microsoft Internet Explorer version 5.2.3 (see Appendix C). The entire experiment code base and supporting files necessary

to run the experiment may be downloaded from http://chil.rice.edu/tambo/dissertation/. The eMacs used CRT displays that measured 43 cm diagonally in a 4:3 aspect ratio. The display resolution was 1024 px by 768 px.

The Phaser interface, as in previous Star Trek experiments, was a single-screen display with controls grouped into clusters (Figure 7a & b). That is, all interface elements used by subjects to complete one trial were visible on-screen at once. The interface elements consist of checkboxes, radio buttons, buttons, a rising thermometer-style gauge to indicate battery Generator level, a clickable horizontal slider for inputing a focus value, a crosshairs with moving target dot, display of time elapsed during the current trial, and feedback display.

Controls used for each subtask appeared within close proximity for each other and within boxes drawn in the interface's background, clearly establishing perceptual grouping of the control objects. Objects that were not used for input, namely the elapsed time and the feedback display, appeared in their own boxes. Interface display background not occluded by some interface element was colored medium-gray. See figures 7a and 7b for the semantically-similar groups and semantic control versions of the Phaser interface.

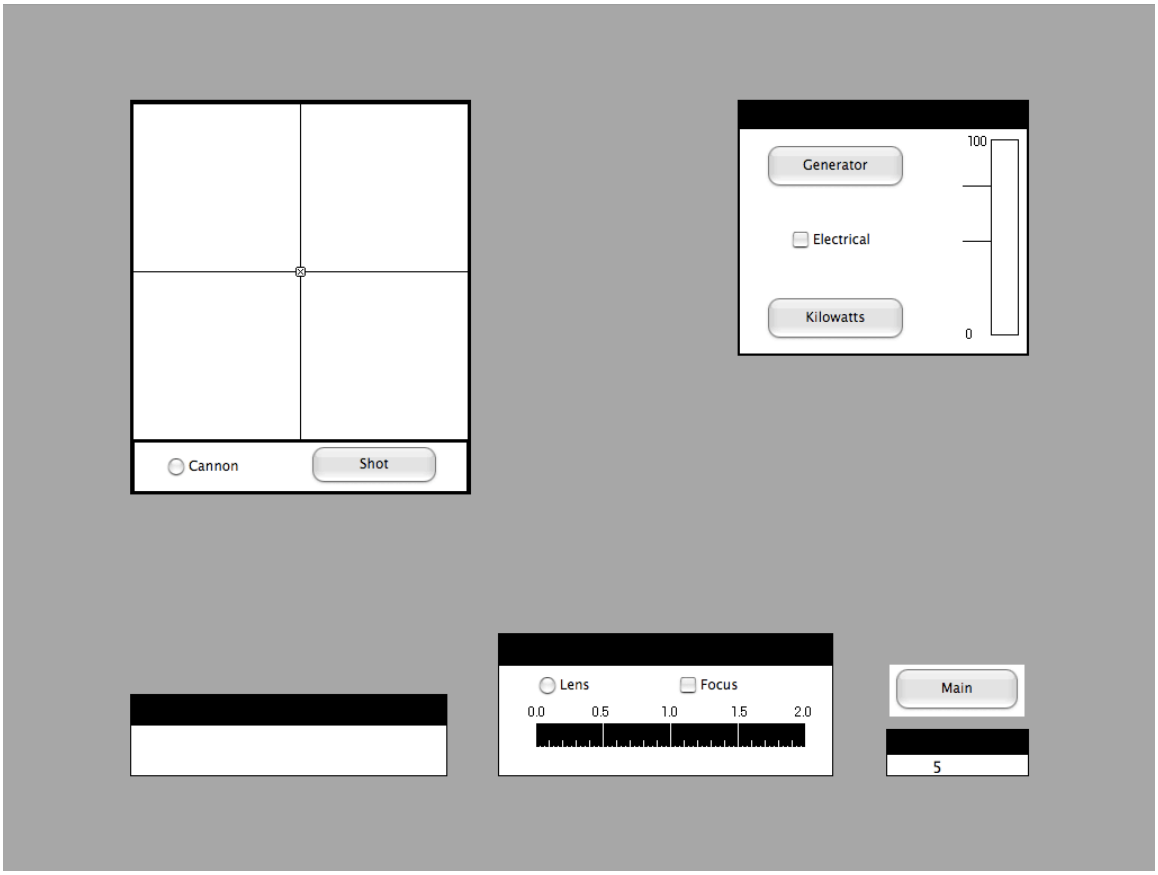

Figure 7a. Phaser interface, semantically-similar control object groups.

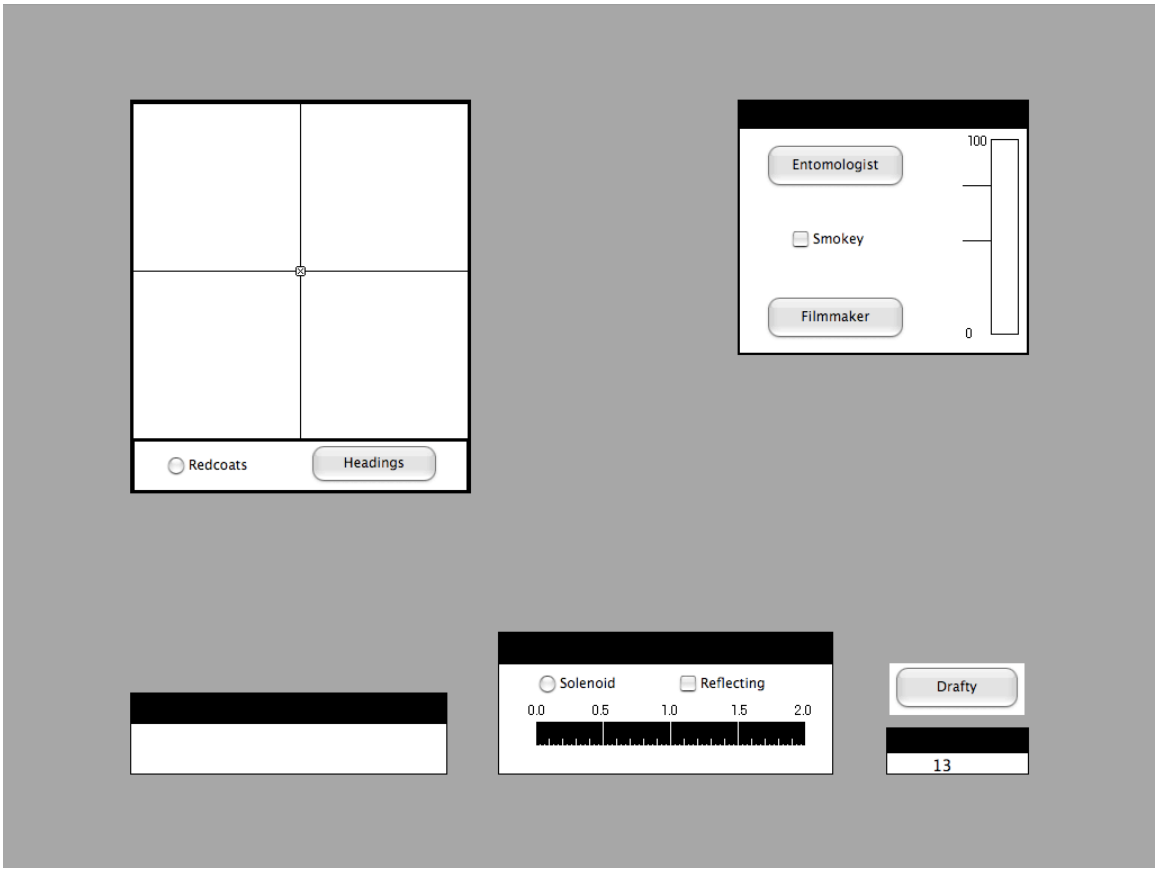

Figure 7b. Phaser interface, semantic control condition.

Except in the semantic control condition, all controls for each subtask, such as charging the battery, carried names in the manuals and in interface labels that were semantically similar to each other but not similar to control names for other subtasks. Latent Semantic Analysis (LSA) established semantic relatedness. LSA is is a general theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text (Landauer & Dumais, 1997; Landauer, Foltz & Laham, 1998). It operates on the principle that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the global similarity of meaning of words and sets of words to each other.

LSA uses singular value decomposition (SVD) to distill a corpus of text into the optimal number of dimensions used to assess similarity ratings. SVD takes as its input a matrix of words and the contexts in which those words appear. Cells represent the transformed raw frequencies with which a given word appeared in a given context, as per Equation 1. The log of the object's frequency + 1 is divided by the object's measured information entropy. The effect of this transformation is to weight each word occurrence directly by an estimate of its importance in the passage and inversely with the degree to which knowing that a word occurs provides information about which passage it appeared in (Landauer, Foltz, & Laham, 1998).

$$\frac{\log(f+1)}{-\sum_{i=1}^{n} p\log(p)} \tag{1}$$

The output from SVD can be used to calculate similarity measures between any combination of two words and/or contexts (paragraphs) compared, which take the form of cosines between the vectors of the factors incorporating those words or contexts. Highly-similar terms might share a cosine of 0.6, while unrelated terms would have a cosine near 0 and dissimilar terms would have a negative cosine.

Tables A1 and A2 in Appendix A list the within-subtasks and between-subtasks cosines for labels used in the semantically-related labels conditions, respectively. Tables A3 and A4 in Appendix A list those values for the semantic control conditions. The within-subtasks cosines measure the semantic similarity of individual labels against other labels in the same subtask while the between-subtasks cosines measure semantic

similarity of all of the terms in one subtask taken as one text against all of the terms in another subtask taken as one text. It is evident from the cosine values in Tables A1 – A4 that in the semantically-related conditions the control labels are indeed similar to each other within subtasks but not between subtasks and that in the semantic control condition the control labels are not semantically similar to each other within- or between-subtasks.

2.1.2.4. Procedure

As in Byrne and Bovair (1997), Chung (2006), and Chung and Byrne (2008), the Phaser task was a sequence of 12 actions performed using an interactive computer interface. Each step consisted of an action such as clicking a button or moving a target onto a crosshairs. The basic Phaser step order followed that used in previous studies except with names changed to manipulate semantic relatedness within and between subtasks. Table 2 lists the steps and subtasks of the basic Phaser task.

Table 2
Phaser Subtasks and Steps

Subtask	Step
Charge Batteries	1. Click “Electrical” checkbox
	2. Click “Generator” button
	3. Wait for power meter to fill to within a pre-determined range, then click “Kilowatts” button
	4. Click “Electrical” checkbox
Set Focus	5. Click “Lens” radio button
	6. Click on the horizontal slider
	7. Check “Focus” checkbox

Table 2
Phaser Subtasks and Steps

Track the Target	8. Click “Cannon” radio button
	9. Click “Shot” button
Fire the Phaser	10. Move target to crosshairs, press spacebar to shoot. If the Romulan vessel was destroyed, then proceed to step 11. If the Romlan vessel was not destroyed, then return to step 1.
	11. Click “Shot” button
	12. Click “Main” button

In the Phaser, the subtasks are to Generator the battery, set the focus, track the target, and shoot. For the version of the Phaser that paused the battery charging subtask, participants clicked the “Generator” button, then performed all steps of the set focus subtask. Only when subjects successfully completed the set focus subtask did the power meter begin to fill. Subjects then waited for the power meter to fill and then clicked “Stop Charging” and then clicked “Electrical.” The trained subtask pausing condition used this Phaser throughout the training and testing sessions. Subjects in the untrained subtask pausing condition trained with the basic Phaser procedure and used it for the first half of their Phaser trials during the test phase. However, halfway through their test session instructions appeared notifying subjects in this condition of the onset and nature of the procedure change. From that point forward they used the paused charging subtask Phaser procedure. The semantic control version of the Phaser used the basic Phaser procedure. The only difference between the basic Phaser and the semantic control Phaser was the set of labels that appeared next to the control objects.

To recap, the basic Phaser procedure followed the steps in Table 2. The semantic control version of the Phaser used the same procedure, but used a different set of labels to provide a control condition where semantic similarity of labels could not be used to group steps into subtasks. The trained intervening subtask version of the Phaser used, throughout the experiment, a Phaser procedure in which subjects completed the first two steps of the Generator batteries subtask, then completed the set focus subtask, then completed the remaining two steps of the Generator batteries subtask. Subjects in the untrained intervening subtask version of the Phaser used the basic Phaser procedure in training and in the first seven trials of the experiment. Before the onset of the eighth Phaser test trial, the experiment displayed a message indicating that the Phaser procedure was to change and what the new Phaser procedure would be – the intervening subtask procedure. Subjects then used the intervening subtask procedure for the remaining seven Phaser trials. A detailed experimenter script provided explicit instructions for the three experimenters to follow when running Experiment 1 and Experiment 2. The experimenter script is included in Appendix B.

2.1.2.4.1. Training Session

When beginning the experiment's training session, participants received instructions aurally from the experimenter (Appendix A). They each also received a packet of manuals: one which described how points are earned during the testing phase, and what the cash prizes were for the top three performers in their experiment group; one which described the "Main Control" interface and provided an overview of the experiment; one for each of the tasks they were to train on in the experiment (e.g., Phaser, Transporter, etc.). After receiving verbal instructions and their training packets, subjects read

instructions from the computer (Appendix A). The written instructions reiterated the experimenter's verbal instructions. When they had finished reading the written instructions, subjects clicked continue to go to Main Control.

Before attempting a task for the first time, subjects read the manual for that task. During the first attempt, subjects kept the manual out for reference while they performed the task. Once they had completed one trial successfully, they returned the manual to the experimenter so that it was no longer available for reference. Participants then continued training as Main Control directed them until they had reached training criterion for that task. Training trials were blocked by trial type so that subjects trained intensively on one task until they had correctly completed four trials. These four correct training trials could be non-consecutive. Upon completion of training, an appointment was set for each subject to return and complete the testing phase four to ten days hence.

2.1.2.4.2. Testing Session

As with the training sessions, testing sessions began with oral and written instruction (Appendix A). After instruction, participants began performing the Star Trek tasks immediately. Once again, Main Control gave trial-by-trial instruction on which task to perform next. Trial types appeared in random order during the testing phase, all in one block. Subjects performed 12 trials of the Phaser task and 8 trials of the Navigation task. Main Control gave non-specific error feedback at the conclusion of each trial, only indicating how many errors had been made during the previous trial. When subjects erred the computer emitted a buzzer sound and the state of the interface did not progress until they performed the correct action.

Subjects also performed a concurrent working memory letter task during the testing session. As in previous studies (Chung, 2006; Chung & Byrne, 2008), its function was to increase working memory load during task performance and thereby elicit a sufficient number of errors to study. Throughout the testing session participants heard randomly ordered letters spoken through the headphones at a rate of one letter every three seconds. A tone presented randomly at intervals ranging from 9 to 45 seconds, accompanied with a response dialog window that popped up on top of the primary Star Trek task window. That is, the letter recall task interrupted the Phaser, Navigation, Main Control, or whatever else it was that the subject was doing at the time. Participants then recalled the last three letters in presentation order and typed them into the response dialog window. Subjects heard a warning buzzer as feedback in the event of incorrect letter string recall.

The instructions displayed on-screen before the beginning of the experiment warned participants in the untrained intervening subtask condition that some of the interface they were to use might change during the experiment. The instructions only stated that a change would take place during the course of the experiment, it did not specify the nature of the change. Before the onset of the eighth trial in the untrained intervening subtask group, a message popped up on screen. The message warned that the Phaser procedure was about to change and it specified the new order of steps to be performed.

As extra incentive for performing to the best of their abilities, all subjects were eligible for cash prizes based on a competitive scoring system. The experiment's test session awarded points based on correct performance, with bonuses for fast performance. Participants earned 25 points for each correct step performed, and a penalty of 50 points for each incorrect action performed. A bonus of 100 additional points was awarded for

each Phaser trial completed in less than 20 seconds. The Navigation task had a 100-point bonus for trials completed in under 10 seconds. Finally, 200 points were deducted from the participant's score for every letter recall trial performed incorrectly. The subject's current score was visible at all times during the testing session in the lower-right corner of the screen. All points counted toward a final score and the three participants with the highest scores received cash prizes of $25 for top score, $15 for second-highest score, or $10 for third-highest score.

2.1.3. Experiment 1 Results

Two dependent measures of interest are error frequencies and step completion times. Error frequencies measured the number of times that a subject made any error at all on a given step divided by the number of presentations of that step. It is important to note that this definition excludes repeated or compound errors. In other words, if on the "Generator" step the subject makes multiple errant clicks before clicking the "Generator" button, the error count for that step would be one. The error count was then divided by the number of times the experiment had the subject perform that step. Furthermore, not all steps furnished meaningful or interesting data. For instance, error frequencies for the "shoot" step were ignored because it was actually composed of a sequence of actions that resulted in one data event, namely multiple key presses to move the target onto the crosshairs and then hitting the space bar to shoot. Step completion times for some steps likewise were not analyzed.

For the procedure-change Phaser, subjects committed 15% more errors on the second "Electrical" step following the change in procedure (Figure 8), interaction contrast of the second "Electrical" step versus all other steps of interest $t(17) = -2.26$, $p = 0.04$. For the three no-change Phasers, the same step induced subjects to produce a step completion time that was 400 ms longer in the intervening subtask Phaser condition relative to the two non-intervening subtask conditions, simple main effect ANOVA with contrast (intervening subtask Phaser versus non-intervening subtask Phaser and semantic control Phaser) $F(2, 62) = 4.45$, $p = 0.02$ (Figure 9). No other effects were reliable, including effects of semantic grouping by label.

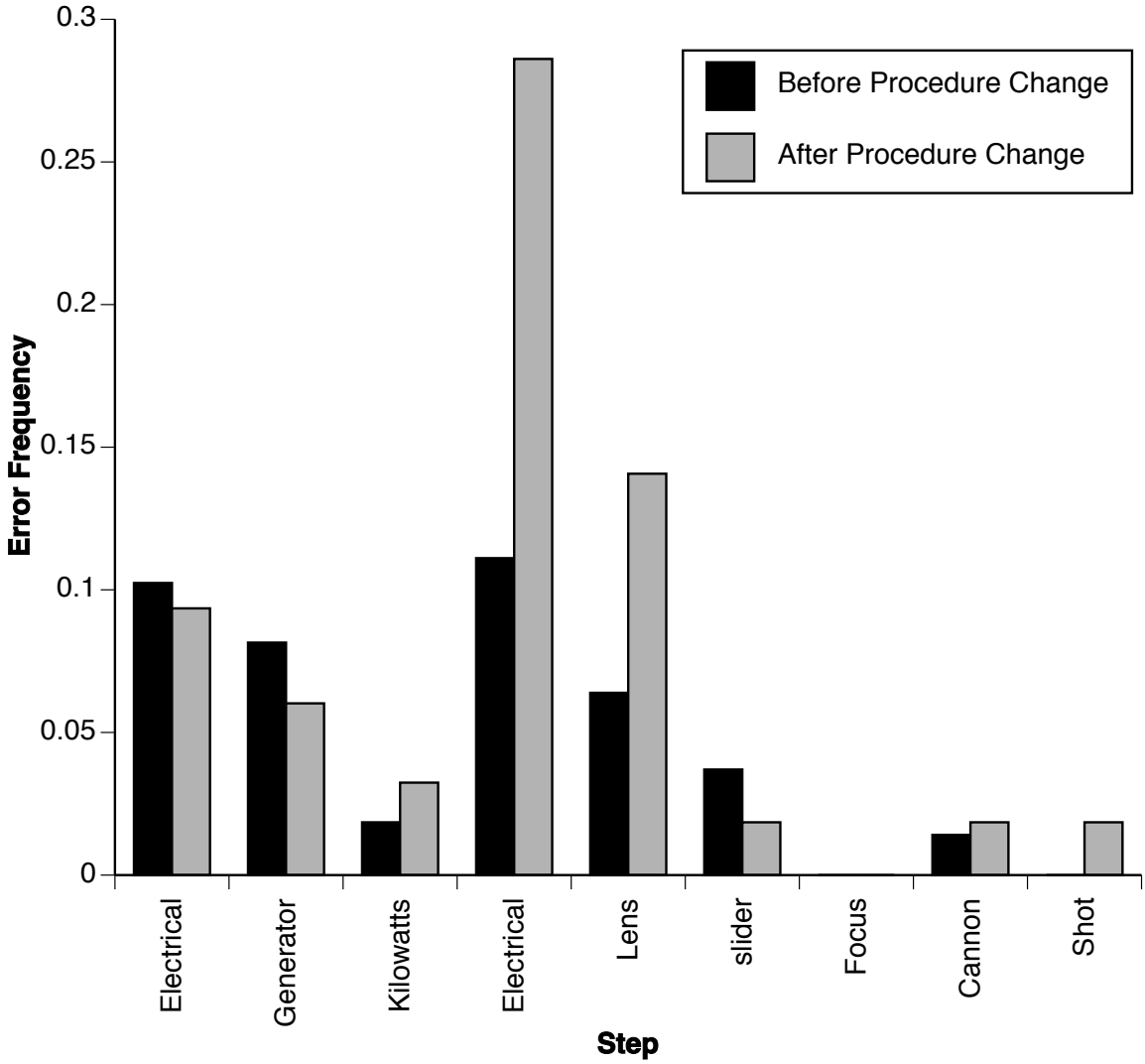

Figure 8. Error frequency in procedure-change Phaser task.

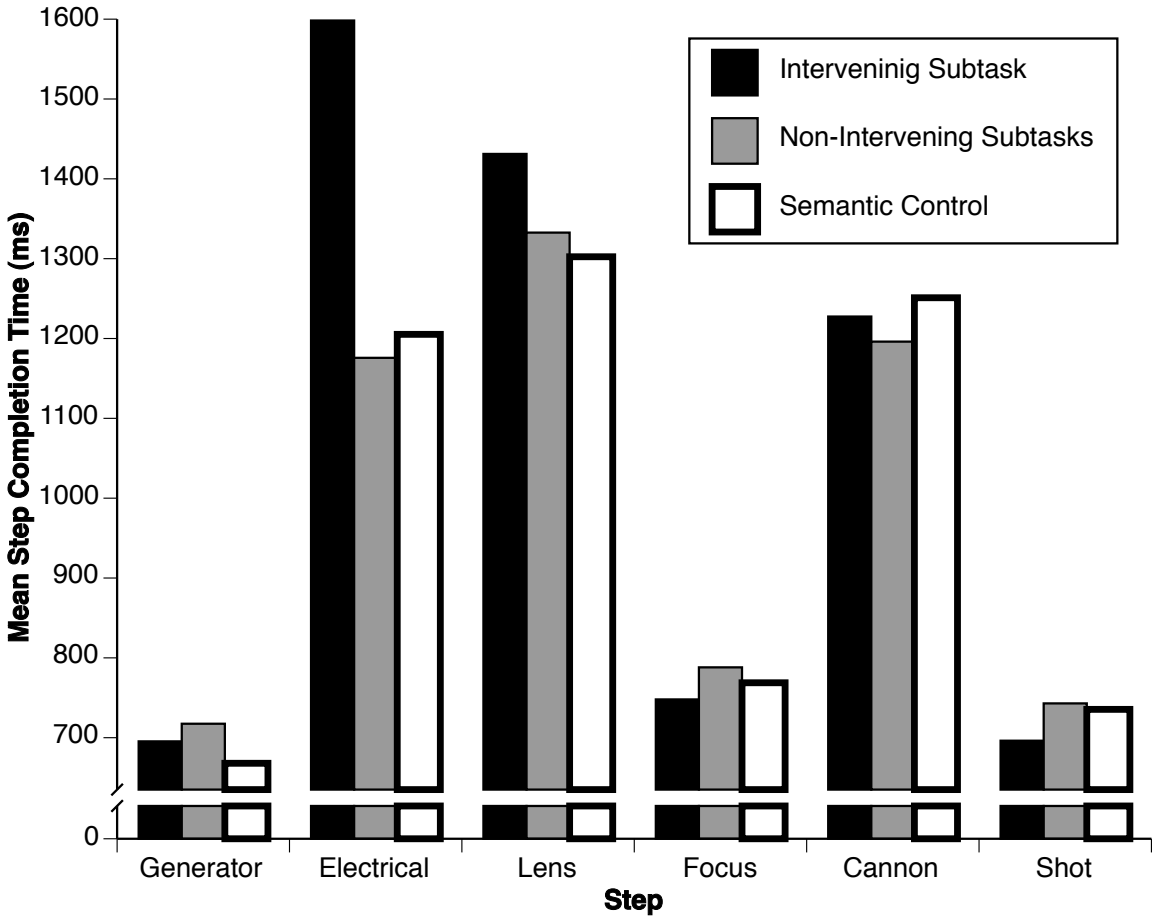

Figure 9. Step completion times for the three no-change Phasers.

Some consideration of error effects that include repeated errors is in order. By considering not only error frequency, but also error severity, some sense of how many attempts it took subjects to recover from errors can be had. The next several analyses consider total error rates per trial type, per condition. The total error rate is equal to the total number of errors committed divided by presentations of task steps. Subsequently this error measure will form the basis of evaluation for an ACT-R model of Experiments 1 and 2.

Figure 10 shows the per-condition trial type error rate obtained from Experiment 1. The no-procedure change conditions did not differ reliably from each other, $F(2, 62) =$

0.40, p = 0.68, but the pre-procedure change Phaser did have a higher error rate than the post-procedure change Phaser, $t(17) = 2.21$, $p = 0.04$. As Figure 11 shows, most of the decrease in error rate came in the postcompletion steps, the second "Electrical" and the second "Shot" steps.

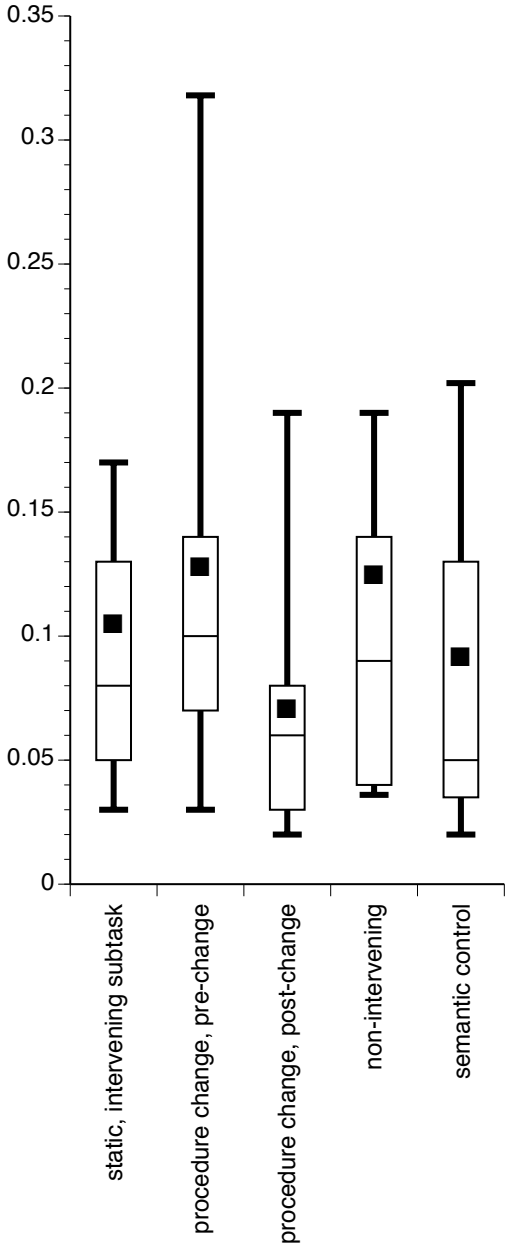

Figure 10. Error rates from each condition of Experiment 1.

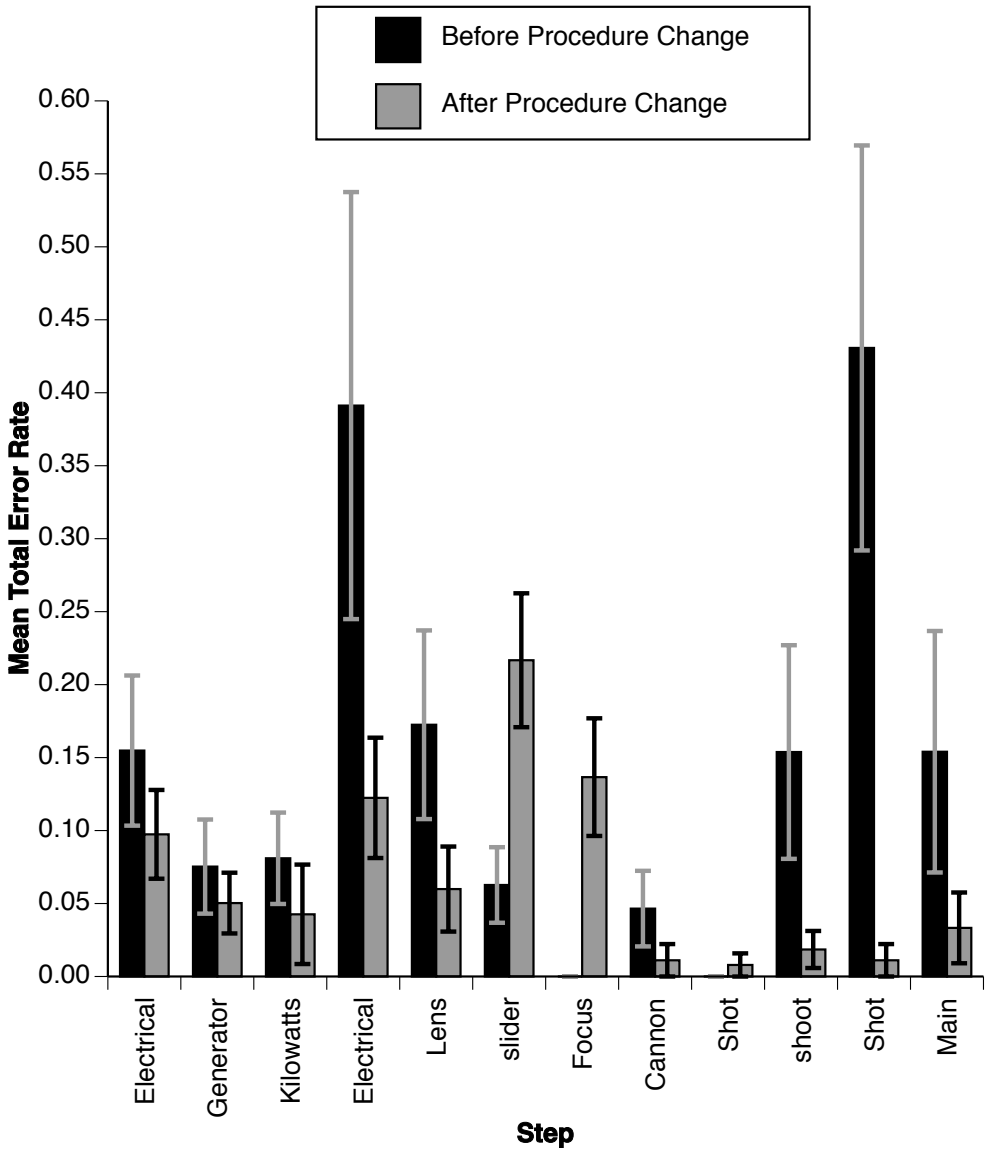

Figure 11. Error rates by step for the Phaser task before procedure change and after procedure change. Error bars represent standard error of the mean.

2.1.4. Experiment 1 Discussion

For the Phaser, in both static and procedure change conditions, intervening the focus subtask in the middle of the Generator subtask seemed to disrupt subjects' ability to perform the second “Electrical” step of the Phaser Generator subtask. It is likely that this step is vulnerable to increased error because it is a postcompletion step for the Generator subtask. The second “Electrical” step is a postcompletion step because once subjects have

stopped charging the battery, the appropriate course of action for participants is to proceed to “Lens” in the non-intervening subtask procedure or “Cannon”, in the case of the intervening subtask procedure.

It may be the case that when the time comes to remember to click “Electrical”, subjects instead mistakenly recall that the next step is “Lens” (or “Cannon”). This is because the two steps are similar in that they follow closely the completion of the main goal of the charging subtask, that is, to complete charging of the battery as signified by clicking “Kilowatts.” In fact, a modeling effort by Chung and Byrne (2008) found that just such a declarative memory similarity combined with a high working memory load can explain the relatively high error rates of postcompletion steps.

It may be that a pronounced postcompletion effect appeared at the second “Electrical” step because of additional memory load requirements imposed by the intervening subtask. Chung and Byrne (2008) found that they could simulate observed postcompletion effects in an ACT-R model by making the similarity between the two chunks that encoded the postcompletion step and its subsequent step relatively high. Because of a combination of that and the high working memory load demands of the task, the chunk encoding the subsequent step would sometimes be retrieved in place of the chunk encoding the postcompletion step, and the model would generate a postcompletion error.

The same mechanism could be at work in the second “Electrical” step. If in the case of this second “Electrical” step, during the non-intervening procedure the memory load may be below a critical threshold. But when the procedure changes, the subject has to use some additional working memory capacity in maintaining an instruction to perform a

different procedure. The extra working memory consumed may be enough to push the subject past the critical threshold and induce postcompletion error.

This pattern of results is problematic for the SRN account because it has no notion of a limited-capacity working memory. Error performance in the SRN stems from degradation of its contextual representation, but the model provides no source for the degradation. Neither does the SRN provide a way to model step similarity and consequently step confusability. Also, because it predicts very rigid task representation the SRN would likely not be able to cope with the procedure change version of the Phaser task. Clearly performance suffered in terms of error frequency for the second "Electrical" step after the Phaser procedure change, yet also the total error rate decreased after the procedure change. The SRN would not have predicted this differential effect.

The complete lack of an effect due to semantic associations in the labels is perhaps not so surprising in light of Tamborello, Chung, & Byrne's (2008) finding that people seem to not actually use labels once they have acquired skill in this type of task. When examined from the perspective of the Soft Constraints Hypothesis (Gray et al., 2006), people are likely not using labels at the skilled stage of performance because reading is relatively slow compared to other retrieval cues available in the Phaser task. These cues include global spatial layout, shape of local button clusters, and relative position within the cluster.

2.2. Experiment 2

2.2.1 Introduction

If subtasks can be delineated by means other than statistical co-occurrence of steps, such as spatial grouping and semantic similarity as manipulated in Experiment 1, then

rearranging the order of steps within a statistically-delineated subtask should not catastrophically impair task performance. To test this second hypothesis, Experiment 2 used the Star Trek Transporter interface. Experiment 2 delineated subtasks by statistical co-occurrence, as per Botvinick and Plaut's (2004) claim. However, the ordering of steps within one of those subtasks will change for one group of subjects half-way through the testing phase. Another condition imposed different orders of subtasks between the Transporter and Jammer tasks.

The purpose of having a second task, the Jammer, on the Transporter interface was to make human performance on the two tasks susceptible to a class of human error termed *mode errors*. A *mode* is a common architecture for grouping several machine configurations under one label. The set of modes in a control system corresponds to a set of unique machine behaviors and the operator can engage those distinct behaviors by switching between modes (Degani, Shafto, & Kirlik, 1999). Action slips can occur when a machine configuration is incorrectly perceived as being in one mode when it is in fact in another mode. In these instances, operators form incorrect intentions and acting upon those incorrect intentions produces a mode error. This is in effect the mechanism by which Cooper and Shallice's CSM generates capture errors – activation from environmental cues to incorrect schemas overwhelms activation from the correct action's schema and the wrong action is selected. Mode errors, then, are a particular case of capture errors induced by a machine's interface when that interface provides the wrong or ambiguous perceptual cues.

When subjects performed the Transporter and Jammer tasks, there were no cues in the environment to remind them which task they were engaged in; they had only the

instruction from Main Control, which disappeared from view once they clicked Main Control's "Transporter" or "Jammer" button to begin that task. In such a task context, it is likely that mode errors may occur when state information about the task falls out of working memory because of working memory demands imposed by the Transporter/ Jammer task and the letter update recall task.

Working memory effects, such as task demands on working memory that contribute to mode errors, could potentially be simulated by the SRN by degrading its contextual representation. But there are other patterns of results that could be problematic for the SRN account. The SRN is very rigid in the action sequences that it outputs. It has no way to reorder steps within a subtask. Any error rate short of total impairment in the procedure change condition would therefore pose a problem to the SRN.

Also, any elevated error rate in the no-procedure change, different-scanner Jammer without similar error increases in that same condition's Transporter task would be difficult for the SRN. All error in the SRN stems from degradation of its contextual representation, which should affect both tasks equally. But a high error rate in one task accompanied by low error rate in another task would indicate another source for error.

Turning to the CSM, results that would be hard for it to explain include catastrophic error rates for the procedure-change Jammer task. CSM posits that the hierarchical structure of task representation combined with the dynamic nature of action selection should allow for some degree of adaptation to the new procedure. Very high error rates in the post-procedure change Jammer would counter-indicate that kind of account of human task representation and action selection.

2.2.2. Method

2.2.2.1. Design

Experiment 2 used the Transporter task as well as an additional task, the Jammer, which was performed on the Transporter's interface. The Jammer is exactly the same task as the Transporter, except in name and as described for the different conditions. All subjects performed both the Transporter and the Jammer tasks.

Experiment 2 incorporated a four-level between-subjects design. Experiment 2 manipulated whether subjects received a Jammer task that was identical to the Transporter task, a Jammer task that had a different sequence of buttons to click in the scanner subtask, a Jammer that changed during the test session from same-Jammer to different-scanner-Jammer, or a Jammer that had its frequency subtask come before the scanner subtask.

The Transporter task was the same for all conditions, but the Jammer changed on a between-subjects' condition basis. The four conditions of Experiment 2 were control, trained step reordering, and untrained step reordering, and subtask reordering. Subjects in the control condition had one scanner subtask to perform, with the ordering of steps the same for both Transporter and Jammer. Subjects in the trained step reordering condition performed the scanner subtask steps in one order for the Transporter and in another order for the Jammer. Subjects in the untrained step reordering condition, like in the Phaser's untrained intervening subtask condition, performed the first half of their test trials using the control scanner ordering for both tasks, then using the changed scanner step ordering for the second half of their Jammer test trials. Subjects in the subtask reordering condition had a fixed order for performing subtasks in both the Jammer and the

Transporter, but in the Jammer, they used a subtask order which differed from that in the Transporter.

2.2.2.2. Materials

Experiment 2 used largely the same materials as Experiment 1, except of course it had its own set of manuals and its own interactive computer interfaces for the Transporter and Jammer tasks. Like the Phaser, the Transporter had a single-screen interactive interface composed of GUI control elements arranged into clusters. The GUI control elements used in the Transporter interface consisted of buttons, radio buttons, check boxes, a scanner bull's eye, two small text fields, a tracking area, a time elapsed display, and a status feedback display. The Jammer task used the same interface device as the Transporter. Figure 12 presents the Transporter/Jammer interface used in Experiment 2.

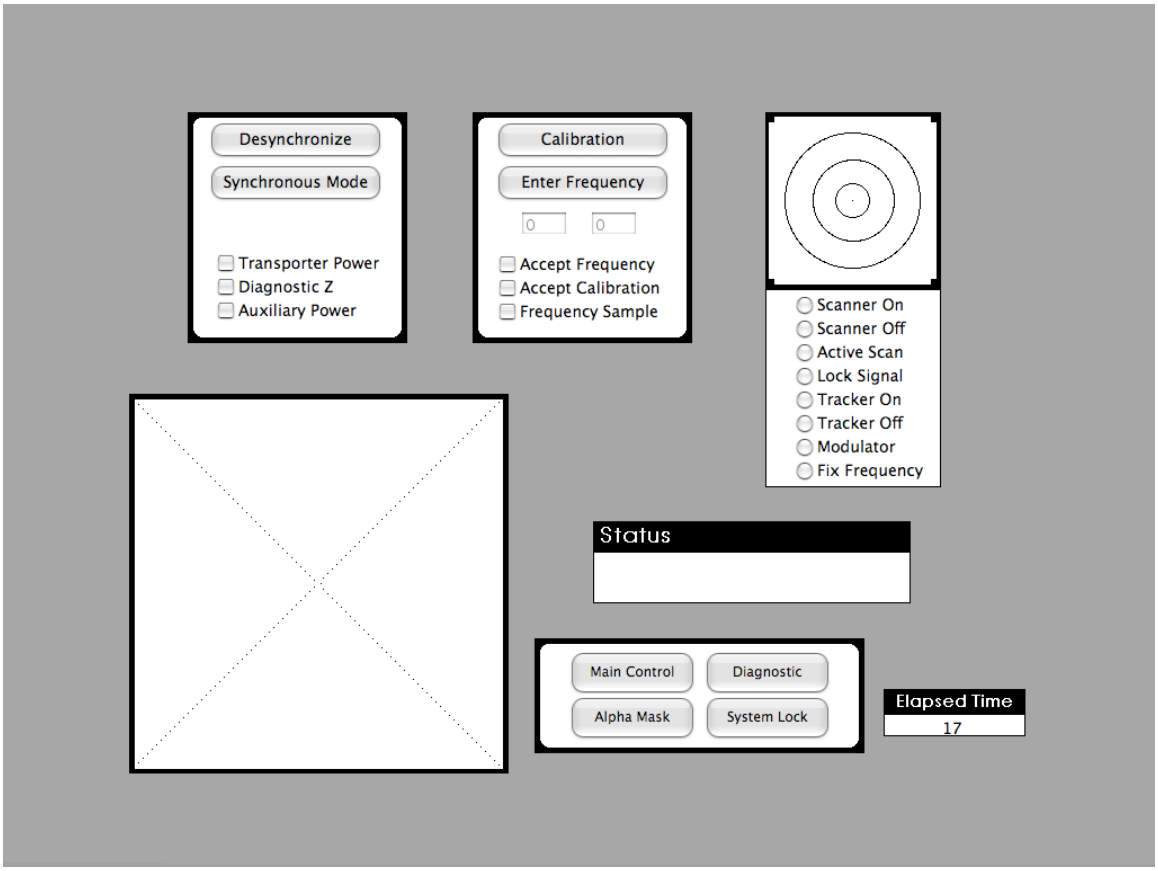

Figure 12. Transporter/Jammer interface.

2.2.2.2.1. Transporter Task

As in previous studies using the Star Trek tasks, the Transporter task was structurally isomorphic with the Phaser task (Byrne, Maurier, Fick, & Chung, 2004; Chung, 2006), but not visually isomorphic. Table 3 lists the steps and subtasks for the Transporter.

Table 3
Transporter Subtasks and Steps

Subtask	Step
Scan for the Homing Signal	1. Click “Scanner On” radio button
	2. Click “Active Scan” radio button
	3. Wait until the scanner homes in on a valid signal, indicated by four scanner dots gradually disappearing, one by one, until only one scanner dot remained. Then click “Lock Signal” radio button
	4. Click “Scanner Off” radio button
Set the Transporter Frequency	5. Click “Enter Frequency” button
	6. Type a two-digit frequency integer into the text field
	7. Check “Accept Fequency” checkbox
Synchronize	8. Check “Transporter Power” checkbox
	9. Click “Synchronous Mode” button
Energize	10. Track the mouse cursor onto the moving target within the tracking area, then click on the target. The task did not advance unless the mouse click fell within the area occupied by the moving target.
	11. Click “Synchronous Mode” button
	12. Click “Main Control” button

In the Transporter, the subgoals were to scan for the homing signal, set the Transporter frequency, synchronize the Transporter with the homing signal, and energize the Transporter. Each time participants performed the Transporter or Jammer during the training session, the experiment first displayed a message indicating in which order to perform the subtasks. For the first three between-subjects conditions, the scanning subtask was always first, followed by either the frequency or synchronization subtasks,

then the remainder of those two subtasks, then the energization subtask. Subjects trained on the Transporter for three trials in each subtask ordering. The training manual for the first thee conditions of the Transporter and Jammer stated that the task could be performed in either subtask order and explained how to do so. During the testing session, subjects were free to perform the Transporter subtasks in whichever order they preferred each time they did a trial.

For the fourth between-subjects condition, the static reordered subtask condition, the experiment did not specify a subtask order for the trial because Transporter and Jammer subtasks were always performed in the same order for each trial, for both Transporter and Jammer. In this condition subjects trained to a criterion of four correct trials, as for the Phaser and the Navigation tasks.

Like the Phaser, the Transporter task also used a probabilistic function to determine whether or not to re-start the trial for the subject. Transporter trials repeated for the same reasons and with the same consequences as the Phaser.

2.2.2.2.2. Jammer Task

The Jammer task was an exact duplicate of the Transporter task with the following exceptions: The first between-subjects group trained and tested using a Jammer that had a different step order for the scanning subtask, “Active Scan,” “Scanner Off,” “Lock Signal,” and “Scanner On.” The second condition used a Jammer that actually was an exact duplicate of the Transporter as a control group. The third condition trained with the Transporter-duplicate Jammer and also used that for the first half of the experiment trials, then switched to the reordered scanner steps Jammer. The Jammer procedure switch used the same method as the Phaser that changed half-way through the experiment.

The fourth condition used a Transporter and Jammer that had only one subtask order. That is, unlike the other versions of the Transporter and Jammer tasks, subjects in this condition were not free to choose whether to perform the frequency subtask before the synchronization subtask or vice versa. The Jammer in this condition had subtasks occur in a different order than the Transporter: Set the Transporter Frequency, Scan for the Homing Signal, Synchronize, and Energize.

2.2.2.3. Procedure

2.2.2.3.1. Training Session

Subjects trained for six trials each on the variable subtask versions of the Jammer and Transporter tasks, three times in each subtask order of each task. The order in which the experiment had subjects perform each subtask order was randomized. Before the training trial started the experiment software displayed a message indicating which subtask order to use during that trial. Subjects thus became familiar with performing the Transporter and Jammer tasks in both possible subtask orders. Subjects in the fourth, static subtask order condition trained to four correct trials each on the Transporter and Jammer.

2.2.2.3.2. Testing Session

The testing procedure for Experiment 2 proceeded as for Experiment 1 but with the following exceptions. Subjects performed 12 trials each of the Transporter and Jammer tasks. As for Experiment 1, instructions displayed on-screen before the beginning of the experiment warned participants in the untrained reordering condition of the change in the procedure they were to perform. The instructions only stated that a change in the procedure would take place, they did not specify the nature of the change. Before the onset of the seventh trial in the untrained reordering group, a message popped-up on

screen. The message warned that the Jammer procedure was to change, and it specified the new order of steps to be performed in the scanning subtask. Scoring remained the same for Experiment 2 except that the 100-point bonus cut-off occurred after 13 seconds for both the Transporter and Jammer. The instructions for the variable subtask order conditions of the Transporter/Jammer reiterated that subjects were free to use whichever subtask order they preferred on each Transporter or Jammer trial.

Since Experiment 1 and Experiment 2 ran concurrently, using the same subjects, every subject in Experiment 1's trained intervening subtask Phaser condition also ran in Experiment 2's Jammer trained reordering Transporter condition. Likewise, Experiment 1's untrained intervening subtask Phaser condition ran with Experiment 2's Transporter and identical Jammer condition; Experiment 1's basic Phaser condition ran with Experiment 2's Jammer untrained reordering Transporter condition; Experiment 1's semantic control Phaser condition ran with Experiment 2's Jammer subtask reordered Transporter condition. Based on prior experience with the Star Trek task paradigm (Chung, 2006; Chung & Byrne, 2008; Byrne, Maurier, Fick, & Chung, 2004) it seemed unlikely that there would be any confounding effects from pairing certain versions of the Phaser with certain versions of the Transporter-Jammer, except perhaps from having a relatively difficult Phaser paired with a relatively difficult Transporter-Jammer. Therefore the experiment conditions were paired so as to prevent any one group of subjects from receiving a relatively more difficult set of tasks.

All subjects participated in a brief post-experiment survey. The survey asked questions about demographics, general computer and internet usage, questions specific to

the experiment, and questions specific to the between-subjects conditions. A copy of the survey is available at http://chil.rice.edu/tambo/dissertation/.

2.2.3. Experiment 2 Results

The same dependent measures were used for Experiment 2 as for Experiment 1. Likewise, some steps were excluded from analysis. Furthermore, any analysis involving the step factor incurred moderate to very large sphericity violations, all Greenhouse-Geisser $\varepsilon < 0.65$. Howell (2002, p. 523) says the literature generally supports the use of Pillai's Trace, so those analyses used the Pillai's Trace MANOVA.

A condition by task by step MANOVA revealed effects of the different-scanner subtask Jammer (Figure 13). For the static different-scanner Jammer task, subjects committed 18% more errors on the second step, “Scanner Off”, than they did for the second step, “Active Scan”, on the same-scanner Jammer task (Figure 14). There was an interaction effect of condition by step Pillai’s Trace = 0.55, $F(8, 29) = 4.47$, $p < 0.01$ and of task by step Pillai’s Trace = 0.57, $F(8, 29) = 4.77$, $p < 0.01$. The condition by task interaction was also reliable, $F(1, 36) = 6.96$, $p = 0.01$. Furthermore, the condition by task by step interaction was reliable, Pillai’s Trace = 0.54, $F(8, 29) = 4.26$, $p < 0.01$ with the effect obviously being driven by the different-scanner Jammer, as further analyses show.

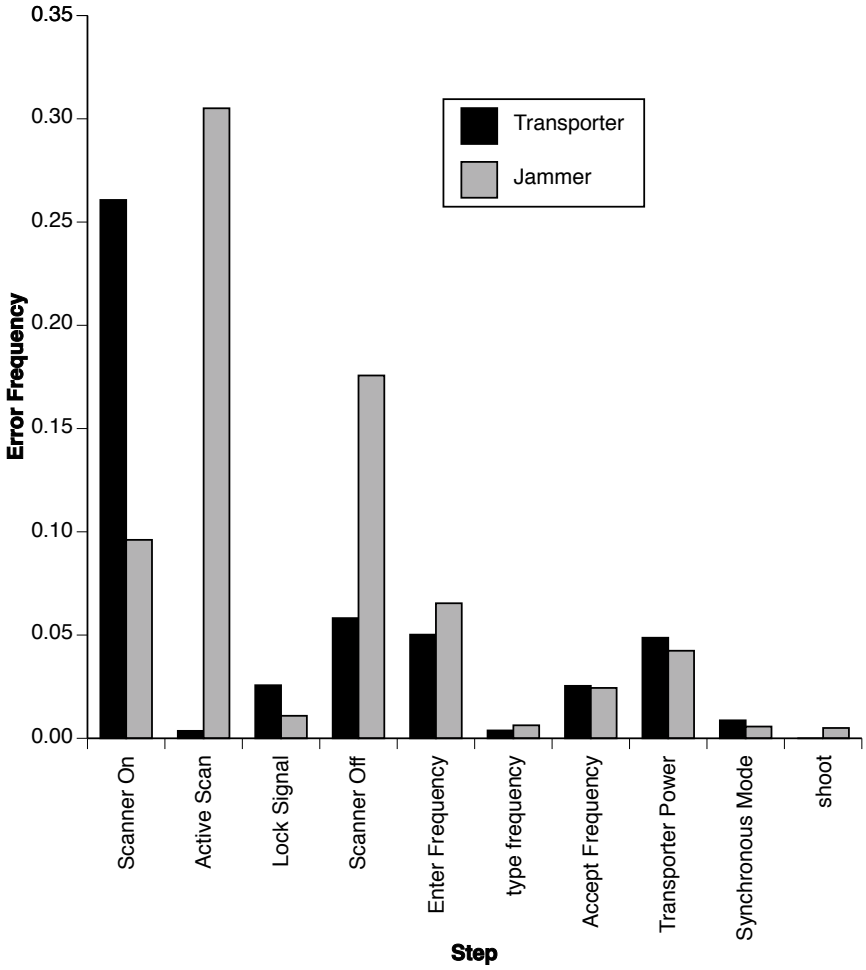

Figure 13. Error frequencies for no-procedure change, different Jammer condition.

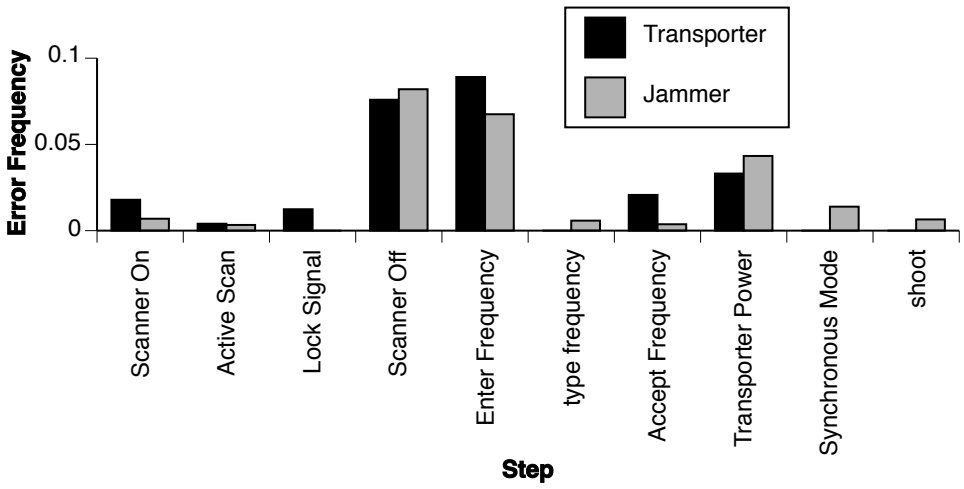

Figure 14. Error frequencies for no-procedure change, same Jammer condition.

Examining only the Jammer task, the simple main effect of condition on error was reliable, $F(1, 36) = 6.52$, $p = 0.02$, and there was an interaction with step, Pillai's Trace = 0.52, $F(8, 29) = 4.00$, $p < 0.01$. A contrast revealed the source of the difference in error rates between the two Jammers to be the second step of the different-Jammer, "Scanner Off," versus the second step of the same-Jammer, "Active Scan," $F(2, 36), = 37.66$, $p < 0.01$. Examining the effect of task at the different-Jammer level of condition, a reliable effect was apparent, $F(1, 19) = 9.60$, $p < 0.01$, and that effect was different for at least one step, Pillai's Trace = 0.74 $F(8, 12) = 4.24$, $p = 0.01$. The step that differed was again the second one for each task, Transporter's "Active Scan" versus Jammer's "Scanner Off", contrast $t(19) = -6.07$, $p < 0.01$. There were no reliable effects of the static different-Jammer and same-Jammer on step response times.

For the procedure change Jammer, there was a main effect of procedure change both on error rate and step completion times for the Jammer task, but no effects on the Transporter task (Figure 15). Change by task by step ANOVAs examined effects on both dependent measures. For error, there was a main effect of procedure change, $F(1, 20) = 11.67$, $p < 0.01$, and an interaction of procedure change with task, $F(1, 20) = 6.16$, $p = 0.02$, such that error rates were worse in the post-change Jammer relative to pre-change Jammer and the Transporter. The effect of task was reliable in the post-change trials, $F(1, 20) = 5.20$, $p = 0.03$, such that after the procedure change, the Jammer had higher rates of error than the Transporter. Furthermore the effect of procedure change on the Jammer task was reliable, $F(1, 20) = 10.17$, $p < 0.01$. A contrast on the step variable revealed that it was the all steps of the scanner subtask together driving the effect of change, t (20) =

-3.03, $p < 0.01$. There was no simple main effect of change on error rates in the Transporter task.

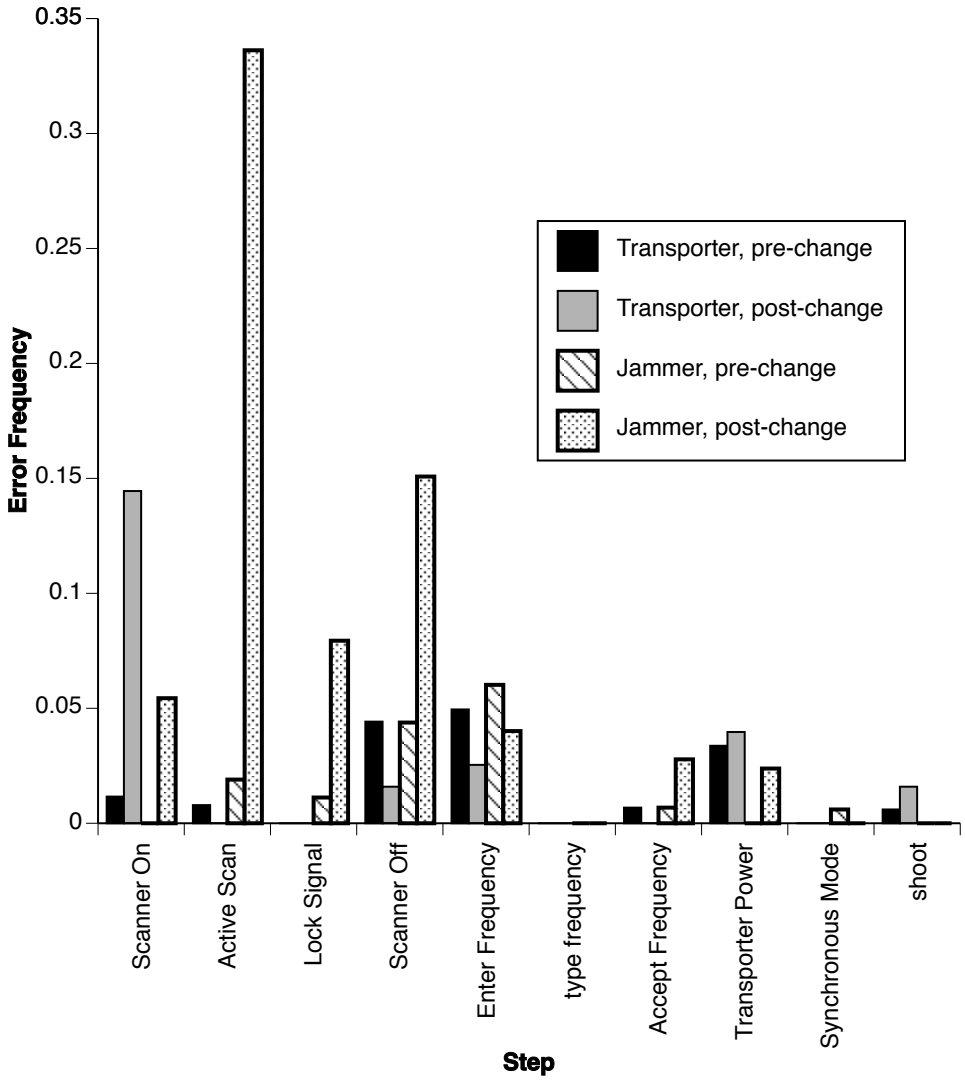

Figure 15. Error frequencies for procedure change Jammer and its accompanying Transporter task.

As regards the Jammer procedure change's effects on step completion times, there was a main effect of task, $F(1, 20) = 24.08$, $p < 0.01$, and an interaction of procedure change and task, $F(1, 20) = 23.73$, $p < 0.01$ (Figure 16). The effect on step completion time of the procedure change was different for some steps, Pillai's Trace = 0.62, $F(7, 14) = 3.24$, $p = 0.03$, as was the effect of task, Pillai's Trace = 0.51, $F(5, 16) = 3.29$, $p = 0.03$.

The task by procedure change by step interaction was also reliable, Pillai's Trace = 0.61, $F(5, 16) = 5.07$, $p < 0.01$. The simple main effect of change for the Jammer task was reliable, $F(1, 20) = 9.48$, $p < 0.01$, the change by step interaction was reliable, Pillai's Trace = 0.58, $F(5, 16) = 4.36$, $p = 0.01$, and that interaction effect was driven by a pre-/post-change difference in step completion times for the steps in the Jammer's scanner subtask, contrast $t(20) = -6.06$, $p < 0.01$. Furthermore, the simple main effect of task in the post-change trials was reliable, indicating higher error rates, post-change, for the Jammer task, $F(1, 20) = 45.20$, $p < 0.01$, and the simple interaction of task by step, post-change, was also reliable, Pillai's Trace = 0.66, $F(5, 16) = 6.10$, $p < 0.01$. There was no simple main effect of change on step completion times in the Transporter task. No effects on error rate or RT were observed for the different subtask order Jammer and its accompanying Transporter task.

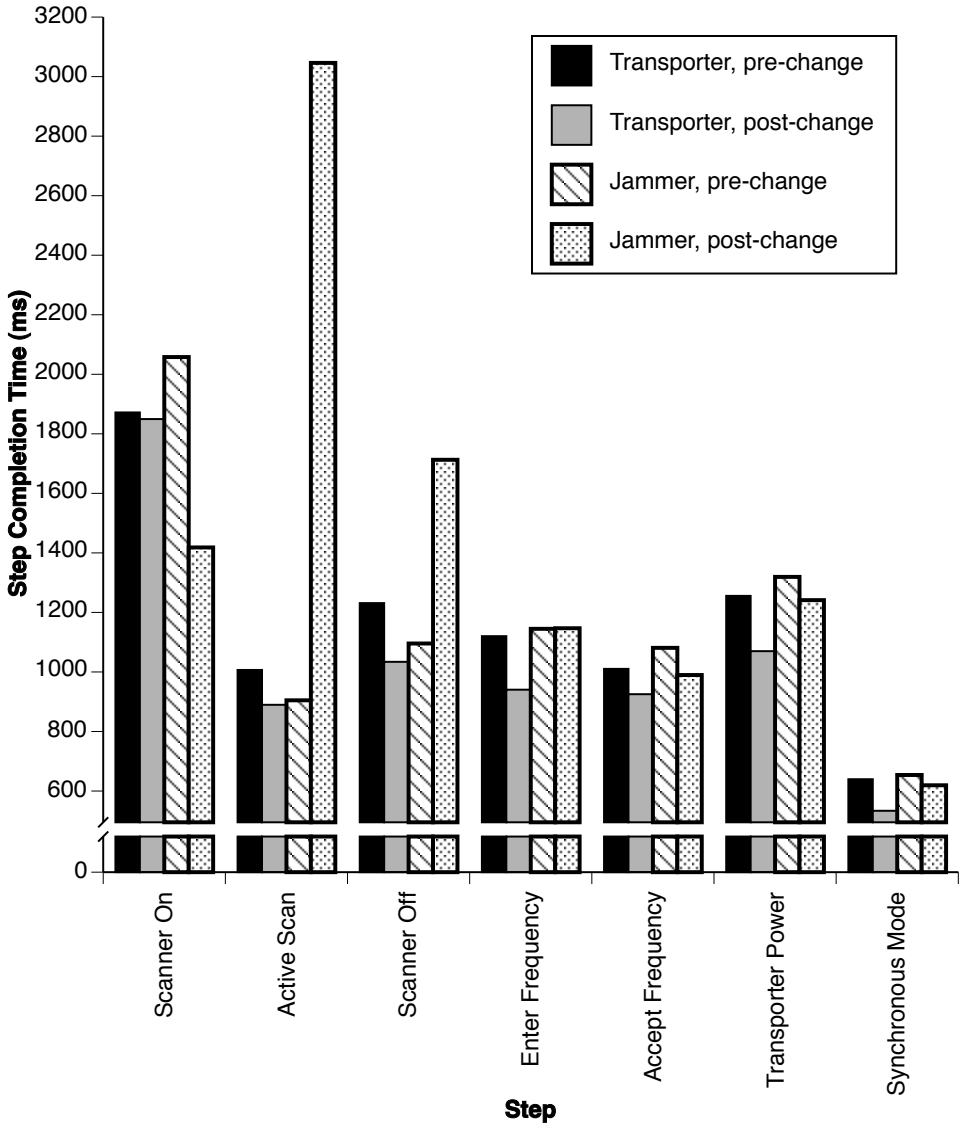

Figure 16. Step completion times for procedure change Jammer and its accompanying Transporter task.

Turning to total error rate, the no-procedure change different-scanner subtask Jammer task elicited a higher error rate from subjects than did the other no-procedure change Jammer conditions, contrast $F(1, 59) = 9.28$, $p < 0.01$. Neither the procedure change Jammer nor any condition of the Transporter task produced effects of error rate. Figure 17 plots Jammer error rates while Figure 18 plots Transporter error rates.

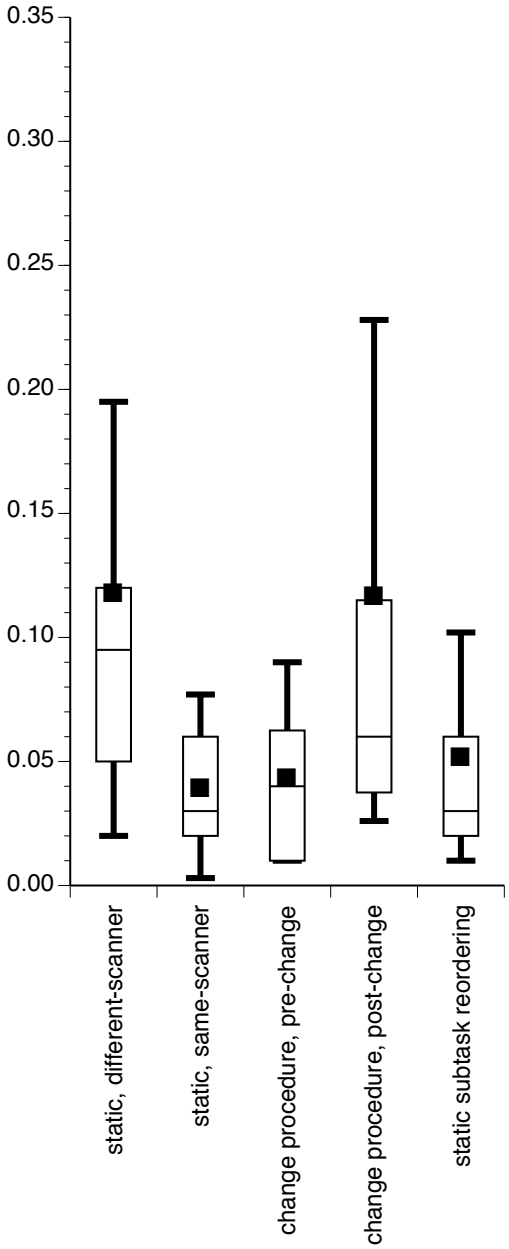

Figure 17. Error rates from the Jammer task of Experiment 2.

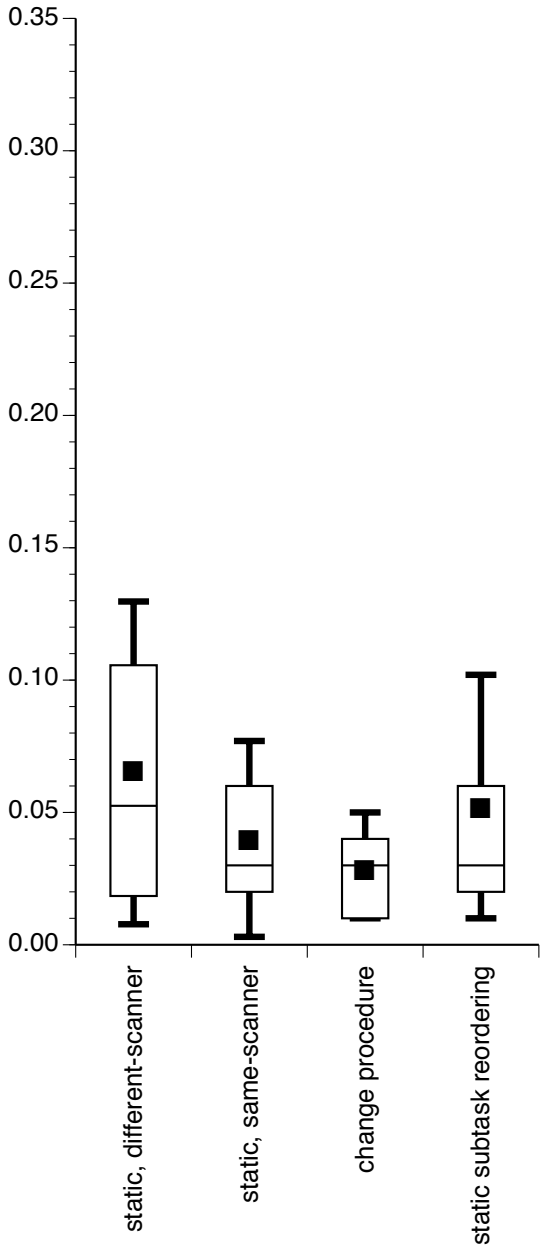

Figure 18. Error rates from the Transporter task of Experiment 2.

Finally, subjects hardly varied the order in which they chose to execute the frequency and power subtasks of the Transporter and Jammer in those conditions in which they had that choice, observed N (frequency, then power) = 1448, observed N (power, then frequency) = 187, expected N = 817.5. It is likely that the effect may have been due either to the spatial arrangement of the Transporter interface or to the presentation order of the

instructions for the frequency and power subtasks in the training manuals. The Transporter interface was laid out such that the frequency controls were always closer to the previously performed subtask's controls, the scanner. Thus subjects may simply have been reaching for the thing closest at hand at the time, a prediction of the Soft Constraints Hypothesis (Gray et al., 2006).

A subsequent experiment tested the two hypotheses by manipulating the Transporter layout such that either the frequency or the power control cluster was closer to the scanner cluster and subjects received training manuals that either presented the frequency subtask before the power subtask or the power subtask before the frequency subtask. The results match the prediction of the Soft Constraints Hypothesis – that is, subjects tended to simply reach for the closest control cluster (Table 4). The Soft Constraints Hypothesis would predict this proximity effect because given the choice of moving the cursor to the near cluster or to the far cluster, the user saves time by going from neighboring cluster to neighboring cluster rather than crisscrossing across the entire screen.

Table 4
Transporter and Jammer Subtask Order Execution Frequencies

Groupings	Frequency, then Power	Power, then Frequency	Expected
The two frequency-closer groups	309	162	235.5
The two power-closer groups	97	288	192.5
The two frequency-first manual groups	185	242	213.5
The two power-first manual groups	221	208	214.5

Note. Frequencies by between-subjects conditions are as follows: frequency-closer layout with frequency-first manual = 216, frequency-closer layout with power-first manual = 255, power-closer layout with frequency-first manual = 211, power-closer layout with power-first manual = 174, expected = 214.

2.2.4. Experiment 2 Discussion

That the error rate was reliably higher in the different Jammer than in the same Jammer is not surprising, but it is surprising that the error rate was not also high in the Transporter task that accompanied the different Jammer. Participants trained with both procedures and performed both procedures on the same interface. Yet somehow Transporter performance did not suffer even though cues indicating the identity of the current task were scant: A brief message and a button label indicated Transporter or Jammer in main control, the sound effect of the shoot action differed between the two tasks, and status messages differed between the two tasks. There was no cue available during the completion of the scanner subtask to remind subjects which of the two tasks they were to perform. So if the effect of change on error rates and RT is evident for the Jammer and not for the Transporter then it seems to imply a relationship between a global

task representation (e.g., “do Transporter”) and a representation of the local action or subtask if subjects are able to keep straight the scanner subtask of the Transporter but not the scanner subtask of the Jammer.

Additionally, because the two tasks share so much information, namely interface objects, the locations of those objects, and procedures aside from the scanner subtask, then it may be that encoding of the two tasks can be accomplished more efficiently by using one representation for both tasks in the places where they overlap. The task representations then will only differ where the tasks differ. For example, it may be that subjects have declarative representations that say in effect, “The Jammer scanner is different from the Transporter scanner. If my goal is to do the Transporter, then start with “Scanner On.” If my goal is to do the Jammer, then start with “Active Scan.” And so perhaps sometimes the task representation structure does in some way mirror the task structure.

Why, then, should the error rates have been higher in the Jammer’s scanner steps than the Transporter’s if the two procedures each have their own differentiated representations for those steps? The worse performance in the Jammer’s scanner could have been due to a semantic confound in the labels. The scanner labels, originally written for the Transporter procedure, were “Scanner On,” “Active Scan,” “Lock Signal,” and “Scanner Off,” in order for the Transporter. The Jammer’s order was “Active Scan,” “Scanner Off,” “Lock Signal,” and “Scanner On.” It may possibly have been confusing having a step indicating an “off” action as the second of four steps in the scanner subtask and having a step indicating an “on” action as the last step. Thus it could be that a differentiated task

representation combined with a confusing order for step names yielded relatively poorer performance for the Jammer.

This combination of differentiated procedure representation and semantic confusion may explain why the subtask-reordered Jammer did not incur any performance hit. In this case, step ordering within subtasks remained the same for the Transporter and Jammer so this Transporter-Jammer pair of tasks would not have been susceptible to confusing confounds like performing an “off” action second in a series of four actions. Yet there was little in the nature of the frequency and scanner subtasks to imply that one should inherently come before the other. This implies that using order-neutral step names within a subtask might extinguish the confusion effect within a subtask, and vice-versa for between-subtasks.

The Phaser interface had no distractor objects while the Transporter interface had 14 of them, and the Transporter task always had the lowest error rates, 0.03 – 0.05 compared to the Phaser's 0.07 – 0.13. So having distractor objects certainly was not sufficient to induce error on its own, though anecdotally it seems that once an error was committed on the Transporter interface, subjects often clicked the distractor objects within a long sequence of clicking wrong things before finally getting back on track. This is likely simply because there were more possible buttons to click at a time when subjects were apparently just clicking randomly until they stumbled upon the right button. Although data from our own lab indicated that adding extraneous buttons does adversely impact performance (Table 1), the distinction appears to be in whether the extraneous buttons were there to begin with or were added halfway through the testing phase. So far it seems

that distractor objects do play a role in action selection, but it is not immediately clear exactly what that role may be.

Experiment 2's results are compatible with the CSM account, but not with the SRN's. The higher error frequencies of the different scanner subtask Jammer relative to the Transporter task or the same scanner subtask Jammer is explainable in terms of schema activations. Top-down activation from the Supervisory Attention System can provide enough activation to the right schemas at the right times to differentiate performance in the Transporter and Jammer. But because the Jammer's scanner steps have semantically order-incongruous names, in order, "Active Scan," "Scanner Off," "Lock Signal," and "Scanner On," the Supervisory Attention System directs some top-down activation to the wrong Jammer step because of semantic confusion from the step labels.

The CSM should be able to account for the procedure change condition of Experiment 2 because of its ability to direct top-down activation to schemas. When the Jammer procedure changes, the CSM's Supervisory Attention System can set goals to perform the steps of the scanner subtask in the new order according to the procedure change instructions. As the time comes to perform each scanner step in the new order, the SAS can transmit enough top-down activation via the vertical threads (see Figure 2) to select the appropriate schema.

While the SRN can also explain the lack of error effects in the Transporter task, it cannot explain the pattern of effects observed in the Jammer. The SRN's context representation should provide enough top-down differentiation of action selection in the two tasks so that Transporter error frequencies would not increase even in the no procedure change different scanner subtask Jammer condition. But again, as in the case of

the procedure change Phaser, the SRN has no way to cope with procedure change in the Jammer.

Additionally, because it is so narrow in scope the SRN would have failed to predict the difference in error frequencies between the no procedure change different scanner subtask Jammer and the no procedure change same scanner subtask Jammer. As far as the SRN is concerned an action is an action and there are no other factors involved in their selection besides the co-occurrence associations that it learned during training.

2.3. Behavioral Studies Discussion

What do the behavioral data have to say about the nature of routine procedure representation and action selection? The lack of error increase for any Phaser step except for the second Electrical – the power subtask's postcompletion step – casts a doubtful shadow on a type of an account that is inflexible, holistic, and purely associative since one discrete step was affected. The results from the static, different subtask order Transporter and Jammer also pose a problem for the SRN account of routine procedure representation. An SRN-type model could perform the two tasks with a low error rate by choosing the first step based on activation from an instruction node, "do Transporter" or "do Jammer." Then when the first subtask is complete, the second subtask is chosen based on the contextual representation. But then those instruction input nodes and the contextual representation together effectively are goals in the CSM sense of the term: they are globally task-relevant representations that direct action selection, particularly in the absence of environmental cues.

For the procedure change Phaser and Jammer, clearly performance suffered, but clearly not to catastrophic extents. True, such changes to procedure are beyond the scope

of SRN, but that is part of the SRN's problem. Procedure change is important. If you drive the same route to work every day, but one day find it blocked by construction, calling in "construction" is not going to appease your boss. People can and do adjust their routine procedures all the time. How those adjustments are made is an important component of the larger question of "How do people represent routine procedural memory?"

How do people adapt representations of routine procedures to the changing circumstances of the world? The SRN is simply too restrictive in scope to be useful. The CSM at least hints at how activation from higher-order motivational and attentional processes can reorder schemas within what usually constitutes one routine behavior and call on schemas from outside that behavior to assist in adjusting to changing circumstances.

An ACT-R model of the behavioral data will need to account for the Phaser's second "Electrical" postcompletion error and account for the scanner steps' error rate in the Jammer and not in the Transporter. The Phaser charging subtask postcompletion error will likely be due to a mechanism like the one modeled by Chung and Byrne (2008), wherein a combination of thinly-spread activation and chunk similarity results in the retrieval of the subsequent step's chunk instead of the correct one. Modeling of the Transporter and Jammer data is likely to be more of a challenge as a solution to the one interface, two tasks, error effect in only one task phenomenon is not readily at hand. But a solution involving one shared basic representation with lower-level representations denoting task divergences would be a good starting point.

3. AN ACT-R MODEL INSPIRED BY THE CONTENTION SCHEDULING MODEL

The ACT-R model operated in a behavioral loop that retrieved an action representation from declarative memory, performed the specified action, and then verified that the action it performed was the correct one. If the action performed was correct, then the model retrieved the next action representation. In the case of an incorrect action, the model entered an error recovery mode.

Procedural memory indicated which action representation to retrieve at any given time. Action representations encoded the visual location and features of the object with which the action was performed as well as the nature of the action (e.g., click, read). The split in what part of the task was represented in declarative memory versus procedural memory was a design decision informed by the Contention Scheduling Model (Cooper & Shallice, 2000). Errors generated by the model were due to misretrievals, which occurred for a variety of reasons. Once an error had been committed, the model had two strategies that it could apply in its attempt to resume correct performance of the current task.

3.1. ACT-R

A brief introduction to ACT-R is in order. ACT-R is a hybrid cognitive architecture. A cognitive architecture is a framework used for creating models of human behavior. The cognitive architecture specifies the resources and constraints that are invariant in human cognition as established by the general consensus of the relevant literature. ACT-R is hybrid in the sense that it combines features of two very different approaches to modeling human cognition: the production system and the association network. Production systems match patterns of conditions to actions to be performed. Knowledge is represented as if-then rules, *productions*, with each production specifying a set of conditions to which it

will match and one or more actions the system will perform when that production is performed, or *fires*. Processing proceeds in a serial fashion with one production firing at any given time.

Association networks arrange knowledge representation structures into nodes that each refer to one or many other nodes. Given an activation source, the association network propagates the activation through the nodes' associations and outputs the node with the highest activation. Processing occurs in parallel with activation spreading through all nodes of the network simultaneously.

ACT-R apportions different types of cognitive processing into modules. Procedural knowledge and actions are effected in the procedural module, which operates as a production system. Declarative knowledge is housed in the declarative module, which operates as an association network. The declarative module's association network is a theory of human declarative memory. A production may specify a *retrieval request*, that is, a command to the declarative module to retrieve a fact.

Other cognitive and perceptual/motor faculties, such as vision and hearing, are implemented in other modules. Each module has a buffer with which it may make a small amount of information, a *chunk*, available to the rest of the system at any one time. Each chunk has one or more *slots*, or places to store references to other chunks or specific values. The procedural module acts as the central coordinator of all of the modules, reading (or sometimes writing) the contents of their buffers and instructing them to perform actions such as to retrieve a declarative memory or to move visual attention.

Two other modules bear mentioning. The goal module holds a special chunk, the *goal chunk*, which keeps track of one's current intentions (Anderson, 2007, pp. 20, 53). The

goal module is particularly important because its slots encode control state information that matches to productions. That way different productions can match to internal states of the model independently of external stimuli. The goal chunk also acts as an activation source for the declarative module.

The other module worth mentioning is the imaginal module. The imaginal module is typically used to store and perform transformations on problem state information (Anderson, 2007, pp. 20, 53). An example of its use is when solving an equation such as *3x - 7 = 5*, it might hold a representation of an intermediate equation such as *3x = 12*.

ACT-R is both a computational psychological theory and a software package which instantiates the theory in a computer program. The program takes as its inputs productions and declarative chunks relevant to a task, a simulated environment in which to perform that task (such as a software GUI), and parameters which adjust computational processes. ACT-R outputs a time-stamped behavior sequence.

ACT-R is important because it brings together accounts of many psychological phenomena into one unifying account that can explain a wide variety behaviors in a wide variety of contexts. ACT-R is computational, meaning that its processes are formally specified and produce qualitative and quantitative characterizations of behavior. Because ACT-R encompasses what is generally agreed to be invariant about human cognition, a model which produces output that resembles human behavior can be used to make inferences about psychological processes responsible for those behaviors.

3.2. Overview of an ACT-R Model of the Star Trek Tasks

The interplay of the procedural and declarative modules, giving ACT-R its hybrid nature, was particularly important in the model that I constructed of the two Star Trek

task experiments. The procedural module performed the actual selection of each action. As task context changed with completion of each step, a production matched to that context and fired, selecting an action to perform. Information necessary for the performance of that action, such as the action type and object to use (e.g., click a button, read a status message) was represented in the declarative module. The interplay came in the procedural module selecting an action and requesting the retrieval of information about that action from the declarative module. Then the procedural module acted according to the action representation retrieved by the declarative module.

Ideally the model would have accounted for quantities at all levels examined in the behavioral data, but modeling human error in 14 separate variations of an experiment paradigm was a significant undertaking. Working top-down worked well for a project where there were many variations on a theme to cover, particularly because people do seem to be cognitive misers and were thus likely to employ the same strategies in all the cases where they could (Gray et al., 2006). Therefore it was important to first address and constrain the basic behavior in the Star Trek task paradigm.

First and at the highest level came issues that were common to all variations on the experiment paradigm: How did people perform the basic task? And more specifically, how were their memory representations of the basic task structured to allow them to perform it? Next came consideration of issues specific to each trial type within each condition, such as: What mechanism generated the postcompletion error on the second "Electrical" step in the Phaser? Why were subjects producing more errors in the Jammer's scanner subtask than the Transporter's? Once the basic model was built,

variations of that model could address the variations of the experiment paradigm that produced these specific error patterns.

Since the CSM's basic structure was modular and hierarchical, it seemed as though a schema network might be built to tackle the Star Trek task. The schema network was programmed in an ACT-R model as productions and declarative chunks. The CSM's approach focused on how familiar action sequences arose out of hierarchical knowledge structures, schemas. It was an approach predicated on the assumption that knowledge structures mirror the structures that they represent, at least as far as action sequences are concerned. These structures were given to ACT-R as productions and declarative chunks representing acquired knowledge of a task and the objects used for that task.

The ACT-R model ran on the same experiment software program used to test human subjects. Since CSM did make allowances for interactions with other cognitive systems, it was in a much better position to take this sort of integrated cognitive modeling approach than the SRN model. Although Cooper and Shallice's system specified how it may interact with other cognitive and perceptual-motor processes, it did not specify how those things operate. However, ACT-R does. The knowledge structures that CSM proposed were given to ACT-R as its acquired knowledge input in the form of declarative memory chunks (including goal chunks) and productions. Goal chunks are simply declarative memory chunks that have a special context: When held in the goal module and made available to the other ACT-R modules via its buffer, it can act as a source of spreading activation to other declarative chunks and productions can match to it.

For the two procedure-change conditions, the ACT-R model needed a way to acquire knowledge of the new procedure as the human subjects did. In the human experiments,

subjects read instructions that appeared on screen before the procedure change took place. How ACT-R can do this, and thus potentially how people can do this, has been addressed by Taatgen, Huss, and Anderson (2008). In brief, their model stored instructions as declarative memory, retrieved them step by step according to the model's productions, and interpreted them and carried them out using other productions. Furthermore, whenever the model failed to retrieve an instruction specifying what action to take it had a body of operators from which it could randomly choose. Taatgen, Huss, and Anderson's model fit data from an experiment examining two types of instructions in a complex aviation task and successfully predicted results from a second experiment. The problem of how to acquire and act upon instructions is non-trivial and has been addressed by such efforts as Taatgen, Huss, and Anderson. Therefore that process was abstracted in the model as explained in the model's methods section.

Two other issues remained outside the scope of the model, semantic manipulation of control labels in Experiment 1 and flexible subtask execution order in Experiment 2. Since there was no effect of semantic manipulation of control labels in the human data, no modeling effort was made to address such experimental manipulations. Also, the issue of flexible execution of the frequency and power subtasks of the Transporter and Jammer tasks was ignored in the model since all effects of subtask order execution seemed to be explainable in terms of Gray et al.'s (2006) Soft Constraints Hypothesis.

The focus of the modeling effort was on human task memory mechanisms that give rise to error. More particularly, why did error rates differ to the degrees that they did between conditions and trial types? Finally, although the ACT-R model generated step completion times as it ran the experiment, those step completion times remained outside

the scope of the modeling project. This was because most of the effects obtained in the behavioral studies were in errors rather than step completion times. This restriction also served to keep the size of the modeling project within tractable limits.

I examined total errors, rather than only non-repeat errors. By examining total errors committed the model was constrained not only in the frequency of non-repeat errors committed, but also constrained in the degree to which the model continued to commit errors before it resumed the correct procedure. Thus it was important that the model incorporated error recovery behavior. The error rates studied therefor came from the total errors committed divided by the number of presentations of that trial type, by condition. Those rates were also be computed for the human participants for the sake of comparison of the model's output to human data, as discussed in the results sections of the two experiments.

3.3. ACT-R Model Methods

3.3.1. Introduction

At its most basic level, the model worked using a behavioral loop inspired by the CSM theory. The model used a goal hierarchy to maintain a contextual representation both of which step within the task it was currently performing and the status of its progress through that step. For each step performed, the model used activation spreading from its goal representation to retrieve from declarative memory information necessary to perform that step. This declarative memory representation included such descriptors of the action as where it was to occur, the action type (e.g., click or read), and visual information about any object that the action required. The model then shifted attention to the visual location specified by the declarative memory representation and performed the

specified action. Finally, the model verified that the action just performed was the correct action and incremented its goal representation to the next step. The model generated errors using the subsymbolic processes of ACT-R's declarative module to occasionally retrieve incorrect action representation chunks from declarative memory. The model recovered from errors by a simple mechanism that either tried to retrieve the originally requested action representation again or tried to retrieve an action representation pertinent to another step.

3.3.2. Influences of the Contention Scheduling Model

The ACT-R model used an explicit, hierarchical goal structure meant to keep track of its progress through the Star Trek Tasks at multiple levels, as in the CSM. Together pattern-matching, schema-like productions and the hierarchical goal representations held in the goal and imaginal buffer chunks formed the basis for the model's structure and behaviors.

Both the goal and imaginal chunks each contained two slots assigned to maintain goal and task context information. The two slots in each chunk simultaneously represented different levels of the model's performance through the task space. For instance, the *state-global* slot of the goal chunk represented the current task step. The *state-local* slot encoded the current phase of the basic behavioral loop. The model cycled through the basic behavioral loop each time it accomplished one of the task steps, with the two goal representations controlling behavior at each level. The imaginal chunks slots represented the top-level context, such as "Do Jammer," and also the current step. The imaginal chunk also had a third slot which encoded the current task interface.

The top-level context representation stored in the imaginal buffer encoded the current task's goal, such as to perform the Transporter or Jammer task. Transporter and Jammer productions matched to the contents of the top-level context slot of the imaginal chunk at steps in the two tasks where they differed from each other (where applicable). For example, the productions that enabled the model to perform the procedure change for the Jammer checked the top-level context slot of the imaginal chunk. When that value indicated the post-procedure change Jammer task, then the model requested the retrieval of the change instruction chunk which specified which action should be performed at that point.

The value of the current step slot of the imaginal chunk was set by the productions that verified correct action performance, and only by the those productions. Since the verify productions only fired in the case of a correctly performed step, the current step slot provided a reliable representation of the model's place in the task – the last correct step performed triggered goal representation of the next step to perform.

The current interface was a reference to the interface type that the model was currently working with, Phaser or Transporter. The current interface had implications for action affordances and therefore what action representation chunks may have applied. Action representation chunks had corresponding interface slots to indicate to which interface they could apply. The interface slot was specified in retrieval requests made for action representation chunks, and the interface value was taken from the chunk in the imaginal buffer. This prevented the model from retrieving action representation chunks that were inappropriate to its current context.

3.3.3. Implementation of a Cognitive Miser and Handling Procedure Change

The model incorporated an economy of task representation that enabled it not only to use one representation for steps shared by tasks, as in the Transporter and Jammer, but also enabled it to adapt old procedure representations to post-change procedures. Because the model used a hierarchical arrangement of discrete goals, as in the CSM account, subgoals and their actions were modular. Thus one subgoal could be a component of more than one supergoal. This meant that for the Jammer task that was identical to the Transporter task, the model's top-level goal for the Jammer task referenced the subgoals for the Transporter task. In other words, the same-Jammer task was composed of Transporter subgoals and action representations.

Discrete, hierarchical goals also enabled the model to cope with procedure changes. When the experiment instructed the model to change procedures, such as going from the non-intervening subtask Phaser to the intervening subtask Phaser, the model simply altered the order in which it executed the relevant subgoals. More detail regarding the model's shared task representations and handling of task procedure changes is included in Appendix E.

3.3.4. Error Generation and Recovery

It was necessary to enable an error recovery mechanism in the model because trials did not end for subjects when they erred, and thus they might commit multiple errors (repeats aside) during the course of one trial. Furthermore, because subjects committed errors in the tasks – indeed, that was the point of the experiments – and because they were not allowed to continue with the task until they had recovered from their errors, error recovery was thus an important part of the Star Trek tasks and a behavior that must have been addressed by the model.

3.3.4.1 Error Generation

ACT-R can model error that is both systematic and stochastic because it is a hybrid cognitive architecture that combines a production system with an associative network system, its declarative module (Anderson et al., 2004). Error can therefore arise as a consequence of either system. Error in the current model was caused by noisy declarative retrieval processes that can sometimes return the wrong representation.

ACT-R's declarative module retrieves the chunk with the highest activation. The fact that several components contribute to a chunk's activation means that the chunk with the highest activation is not necessarily the one referenced in the retrieval request specification. Equation 2 describes how activation is computed for any given chunk *i* at the time that it is being evaluated for retrieval. Activation is the sum of a base level activation term, a spreading activation term, a partial-matching term, and noise.

$$A_i = B_i + S_i + P_i + \varepsilon_i \tag{2}$$

Spreading activation is essentially the total amount of activation source available divided by the number of occupied slots in the source chunk (here, the goal chunk) times, for each slot value that refers to the target chunk, the strength of association to the target chunk. Chung and Byrne (2008) simulated high working memory loads by occupying three slots of the goal chunk with dummy chunks that "stole" activation available for the retrieval request since they were in the goal chunk but did not refer to the target declarative chunk. Those three slots acted like the state information used for the working memory letter recall task. Chung and Byrne's technique was adopted for this ACT-R model.

The model ran with partial matching enabled. This means that when ACT-R attempted a retrieval, if a chunk in a slot specified in the retrieval request did not match the chunk in the corresponding slot of the chunk under consideration, then ACT-R calculated a mismatch penalty. The idea is that chunks not matching the retrieval specification get penalized by some amount of activation proportional to how dissimilar they are to the retrieval specification. That mismatch penalty is equal to the similarity between the two chunks times the match scale parameter. The match scale parameter was set to 1.8 as an empirical fit.

Retrieval transient noise is a feature of the ACT-R theory based on decades of cognitive psychological research and as such is meant to embody a basic property of human declarative memory. ACT-R can be configured to run with some user-specified amount of noise added to the computation of chunk activation in the retrieval process. Noise is a transient component which is computed each time a retrieval request is made and it is generated from a logistic distribution with a mean of 0 and a standard deviation as specified by the activation noise parameter. The activation noise parameter of this project's model was set to 0.23, which is in a fairly conventional range for ACT-R models (Chung & Byrne, 2008).

Other global parameters set for this model controlled aspects of chunk retrieval. Maximum associative strength, the maximum strength of association between two chunks, was set to 2. Retrieval threshold was set to -2 which meant that virtually all retrieval requests would result in something being retrieved. Appendix F contains several tables detailing global and chunk parameters, their values, and reasons for setting them to their values.

Parameters for many individual chunks were fit empirically. For example, some action representation chunk base levels, *B* in Equation 2, were set manually. While base levels defaulted to 0, the two postcompletion steps of the Phaser had lower base levels. The relatively lower base levels of these and other chunks decreased the likelihood that these action representation chunks would be retrieved upon request, thereby increasing the likelihood of error at these steps to rates that approximated those found in the human data. The action representation chunk for the second instance of "Electrical" was set to -1.2 and the action representation chunk encoding the second "Shot" step was set to -1.1. The base levels for three of the Transporter interface's action representation chunks, "Scanner On", "Scanner Off", and the second instance of "Synchronous Mode", were all set to -0.4.

Chunk similarity defaulted to the maximum difference, which was set to -10. This was useful as a way to prevent chunks appropriate to one task interface from being retrieved while the model was performing a task with another interface. That is, like having the "interface" slots in the imaginal chunk and the action representation chunks, it prevented chunks encoding Transporter steps from being retrieved during a Phaser trial. Typically, chunks within a subtask had similarities set to -0.95 while chunks in different subtasks within the same task had similarities of -1. Certain chunks encoding steps that seemed to be confused with each other by the subjects, such as in a postcompletion step, had higher similarity. For example, because clicking "Main" is appropriate when the task is complete and the participants know that the target was destroyed (therefore completing the main task), the second "Shot" and the "Main" chunks are given a higher-than default similarity, meaning that when one is requested, the other can sometimes be retrieved in its

place. This was even more likely when working memory load is high (Chung & Byrne, 2008). High chunk similarities can also encode things like a semantic confusion effect of doing "Scanner Off" second, after “Active Scan” and before “Scanner On” in the scanner subtask. These chunk similarities had values as high as -0.7.

For the particular change instruction chunk that was supposed to be retrieved by association with a particular action representation chunk, the strength of association ranged from 1 to 2. For change instruction chunks not to be retrieved by a particular action representation chunk, that value ranged from -0.5 to 0.5. Again, all non-default chunk parameters were fit empirically.

Probability of generating an error became higher when two chunks must be retrieved for the completion of one step, as in the different-scanner Jammer task. The model could retrieve the wrong change instruction chunk, in which case it would have an instruction indicating the performance of a wrong step. Of course, if the model had retrieved the correct change instruction in the first place it could still misretrieve the action representation chunk indicated by the change instruction chunk.

3.3.4.2. Error Recovery

Neither Cooper and Shallice (2000) nor Botvinick and Plaut (2004) discussed error recovery. But recovery from errors is an important component behavior of executing routine procedures because life does not stop when you reach for the cream having meant to reach for the sugar as it did in these two models of coffee making. A model implemented in the CSM theory could possess schemas that encode error recovery procedures. Botvinick and Plaut, however, restricted the scope of their modeling effort so much that they left out any consideration of error recovery in routine procedures. Their

model only generated the initial selection of the routine actions. Furthermore, in order for the SRN to be effective at error recovery it would have had to learn correct action outputs for every possible error state context.

The experiment emitted a buzzer sound whenever subjects erred in the Star Trek tasks. The model used this feedback to trigger initiation of an error recovery procedure. There were two error recovery strategies that it could perform: One was to simply try again to retrieve the action representation chunk indicated by the step state information in the goal chunk. Another strategy was to instead try to retrieve some other action representation chunk that was not the one indicated by the step state information in the goal chunk (James McClelland, personal communication, October 20, 2008). In either case, once the model had retrieved an action representation chunk it resumed the basic behavioral loop.

Additional failures could result from other circumstances. If the model simply failed to retrieve any action representation chunk it could likewise opt to make the same retrieval request again or try to retrieve some other action representation chunk. If the model committed an error while acting on a change instruction, it could try again to retrieve a change instruction chunk using problem state information encoded in the current step slot of the imaginal chunk. The model could use the value of that slot to attempt again to retrieve the appropriate change instruction. Additionally, if the model was trying to recover from an error it could try to retrieve an applicable action representation chunk at random.

The ubiquity of the “if error, try retrieval” procedure, in one form or another, acted as a sort of catch-all mechanism to ensure that the model did not stop running until it had

finished its task. Additionally, having the model retrieve random, unspecified (within the constraints of being appropriate to the current task interface) action representation chunks served to generate a sort of "lost" behavior wherein the model simulated a long chain of errant and seemingly random clicks until it had hit upon the correct action. This type of random clicking behavior has often been informally observed in the human data from the Star Trek tasks.

3.3.4.3. Error Recovery Strategy Choice

Both Experiments 1 and 2 induced time pressure to perform as the experiment awarded significant bonus points for finishing trials of each task within certain time limits. ACT-R's utility learning mechanism already takes into account the passage of time, but not time pressures induced by the task such as in Experiments 1 and 2. Time pressure mattered for deciding whether to try again to retrieve an action representation chunk indicated by goal information within the goal chunk or to abandon that goal information and try instead to retrieve a random action representation chunk and perform its indicated action. Fundamentally the model tried to pick the error recovery strategy with the shortest expected time to lead to a correct step execution. Both error recovery strategies were encoded by productions that competed with each other, with the higher-utility production usually winning selection. ACT-R learns the utility of each production according to Equation 3.

$$U_i(n) = U_i(n-1) + \alpha\left[R_i(n) - U_i(n-1)\right] \quad (3)$$

Production utility learning occurs when a reward is triggered, and all productions that have fired since the last reward are updated. The effective reward of a production *i* is the

reward value received at time *n* minus the time since the selection of production *i* (Anderson, 2007, pp. 160 – 1). The learning rate, α, was left at ACT-R's default, 0.2. Experiment reward points for subjects were directly translated into reward units for ACT-R such that the model received 25 units of reward for every step that the model completed correctly and -50 units of reward for every error that it committed. Constant time pressure worked by triggering a reward of -5 reward units for every second that elapsed during a trial. Thus the model's choice of error recovery strategy should have been sensitive to the passage of time and to reward signals emanating from the task interface in the form of the presence or absence of the warning buzzer as a consequence of each action performed by the model. However, it should be noted that this particular model has not been evaluated against another similar model that did not implement a constant time pressure mechanism.

3.4. ACT-R Model Results

Examination of results of the present model will focus on the per-condition trial type error rate measure developed from the behavioral studies. This measure will be used to assess what the experiment and model results have to say about how people structure task representations and how those representations contribute to human error. Most of the effects observed in the two behavioral studies were in some measure of error and not in step completion times, and therefore the most meaningful pattern of results between per-condition trial types of the model would be measured by relative error rates.

Model development focused on per-condition trial type error rates, rather than only non-repeat error frequencies. This measure of error ensured that the model's error recovery procedures allowed the model to recover within a range of number of mistakes

comparable to humans because it included the degree to which the model selected random, wrong actions while trying to recover from the initial error. It is an easy way to compare the degree of random action selecting in the model's output to the subjects'. Error recovery is an important component of the task and will be captured implicitly by setting total error rate, rather than non-repeat error, as the modeling criterion dependent measure.

One separate model ran for each condition of the two experiments. All models were identical to each other to the extent that their experimental conditions were identical. The models differed from each other only as a function of their differing tasks. For example, in simulating the postcompletion step of the Phaser's charging subtask, chunk similarities were high for the second "Electrical" chunk and whatever chunk encoded the step that came after it. That step was "Lens" in the non-intervening subtask version of the Phaser and it was "Cannon" in the intervening subtask Phaser.

All four models used the same values for global parameters. Declarative memory consisted of 48 chunks of 7 chunk types, not including chunks that encoded change instructions because they were not applicable to all experiment conditions. Except as noted previously, chunk parameters remained identical across the four models, and for every chunk in every model this included parameters controlling base levels, chunk similarities, and strengths of association.

Productions remained identical across the four models to the extent that procedures remained identical across the four experiment conditions. All four models had identical productions encoding the basic behavioral loop, error recovery procedures, and procedures for acting on change instructions, where applicable. The only productions to

vary between the models were the productions that selected the next actions to perform. Although, notably, these productions were identical for the Transporter and Jammer tasks to the extent that the Transporter and Jammer tasks were identical.

Each model ran 400 times on its version of the experiment, though data from some model runs was discarded because of software crashes. Since the ACT-R models were stochastic processes, 400 runs should be a more than sufficient size to obtain stable means for error quantities. Table 5 enumerates experiment conditions and how many model runs in each condition contributed data to analyses. Finally, it seems like a reasonable target criteria for assessing a model that spans all 14 per-condition trial types of two experiments would be to generate error rates that fall between the 25th and 75th percentiles of the human error rates.

Table 5
Experiment Conditions and Model Runs

Experiment 1 Condition	Experiment 2 Condition	Runs N
No procedure change, intervening subtask Phaser	No procedure change, different scanner subtask Jammer	400
Procedure change Phaser	Identical Jammer	386
No procedure change, non-intervening subtask Phaser	Procedure change Jammer	392
Semantic control Phaser	Jammer, different subtask order	397

Figures 19 – 21 compare mean error rates between human data and the model's results. The error rate for any given trial type in a condition was equal to the sum of all errors, including repeats, across all twelve steps divided by the sum of all step presentations, or instances of opportunity for error. The model deviated from the subject means by a maximum per-condition trial type error proportion of 0.033 and by a minimum of 0.0007, with a median deviation of 0.0166.

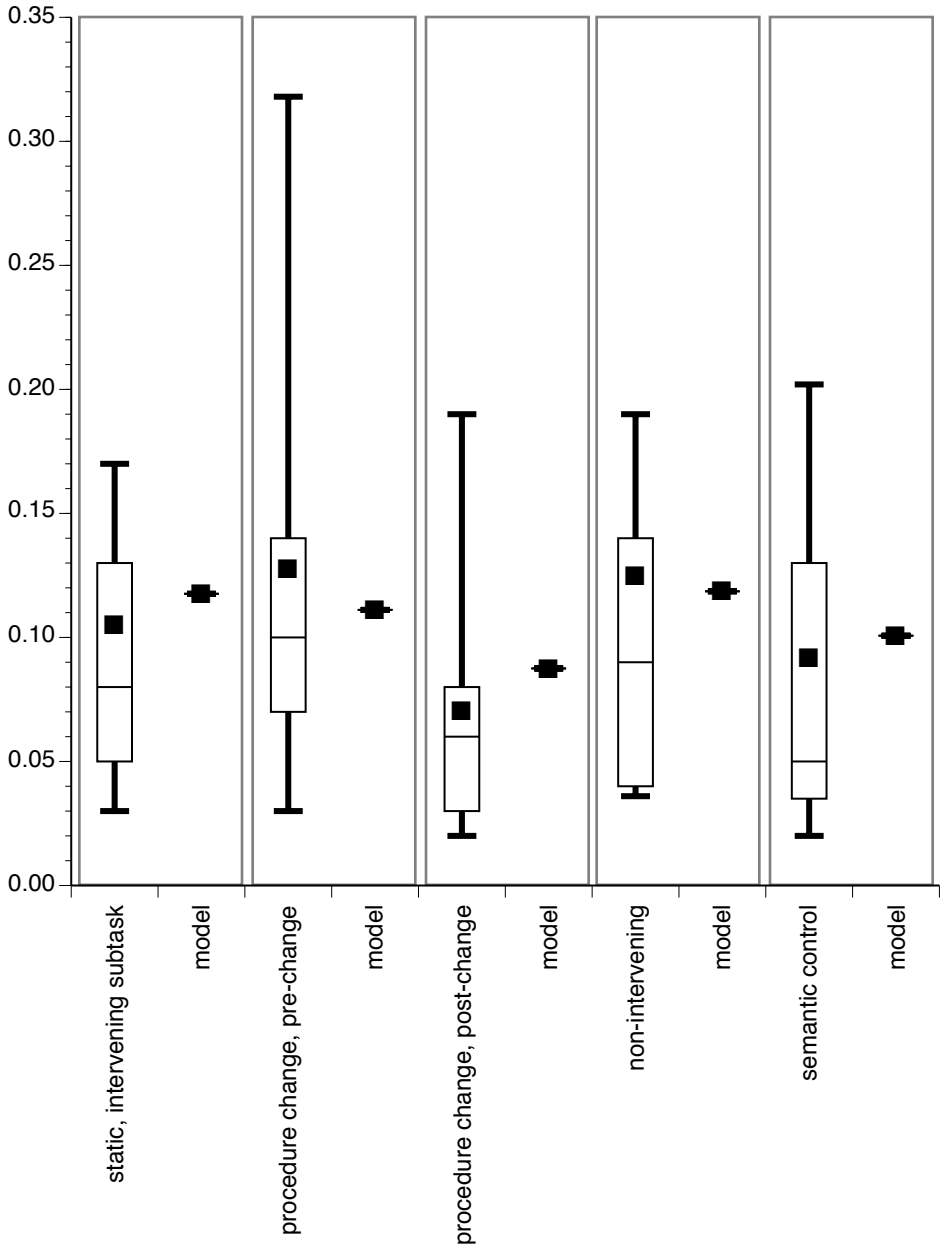

Figure 19. Phaser error rate, by condition of Experiment 1. Box plots represent human data distributions with means depicted as dots within boxes. Model means are depicted as adjacent, lone dots. R^2 for model prediction of human data = 0.755.

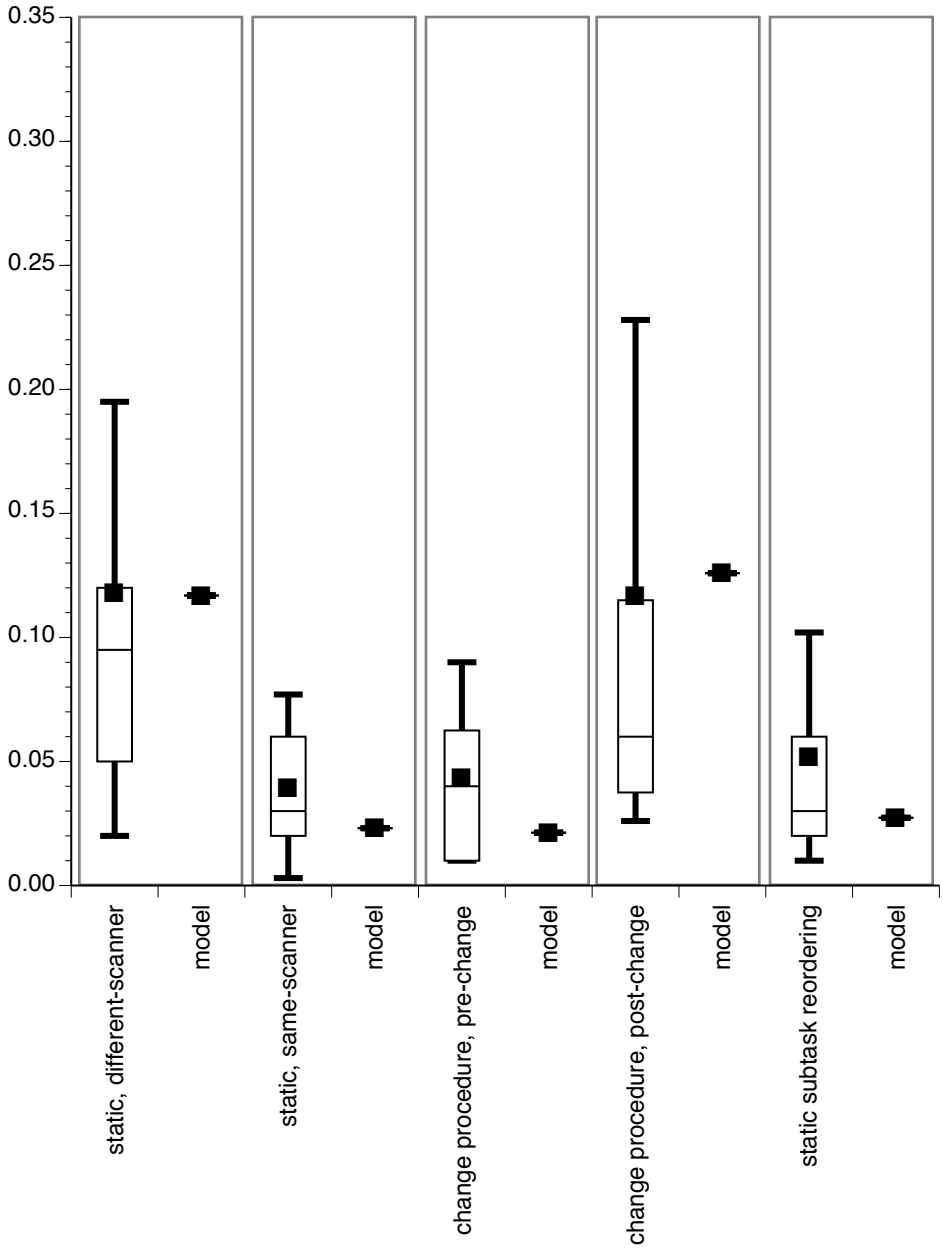

Figure 20. Jammer error rate, by condition of Experiment 2. Box plots represent human data distributions with means depicted as dots within boxes. Model means are depicted as adjacent, lone dots. R^2 for model prediction of human data = 0.989.

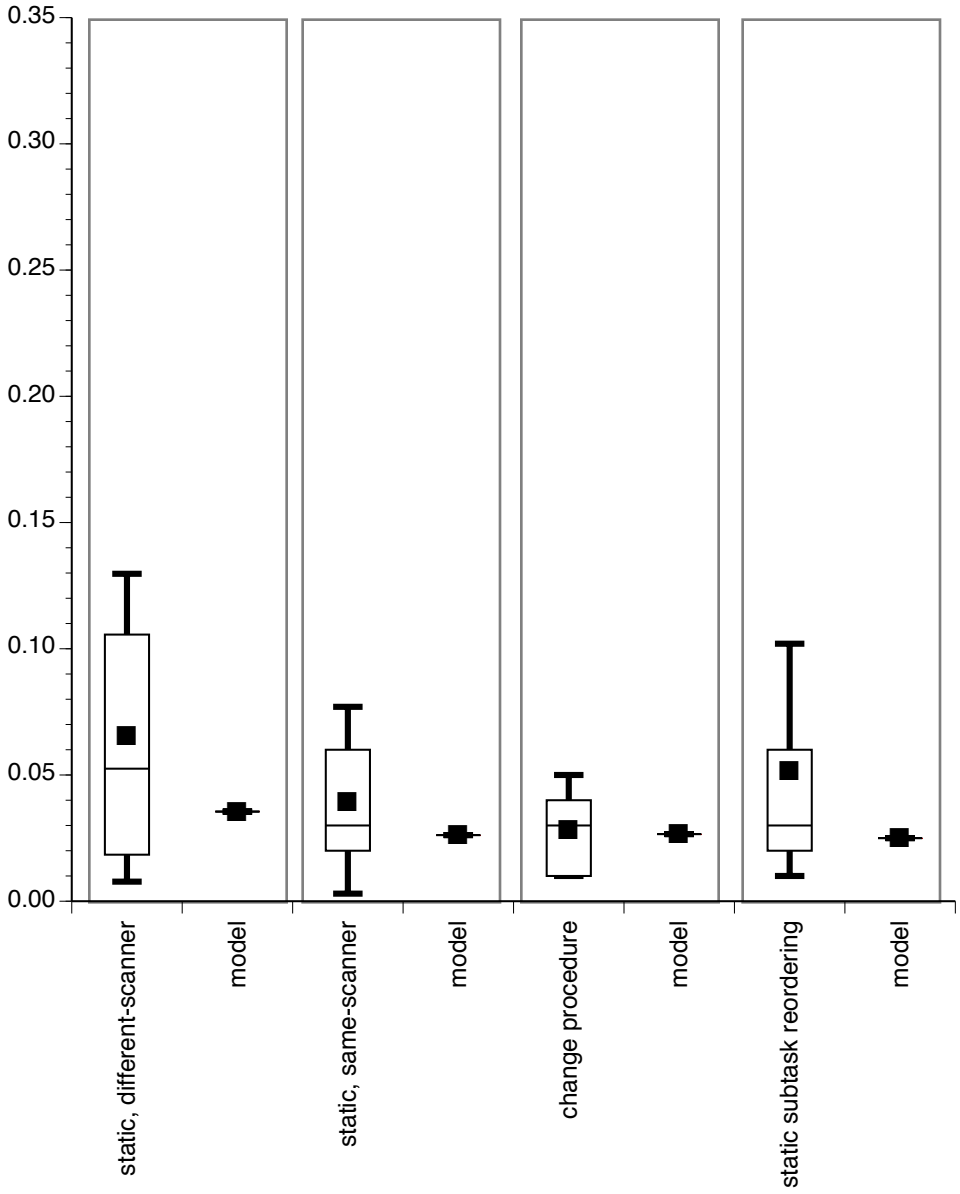

Figure 21. Transporter error rate, by condition of Experiment 2. Box plots represent human data distributions with means depicted as dots within boxes. Model means are depicted as adjacent, lone dots. R^2 for model prediction of human data = 0.278.

3.5. ACT-R Model Discussion

3.5.1 Summary

What aspects of the model were important for its ability to vary between error proportions of roughly 0.02 to 0.12 in a wide variety of experimental conditions? What do each of those aspects have to say about human task representations and how they give rise to the qualities and quantities of error observed? Most of all, how was it that essentially the same model was able to match error performance in 14 trial types across 4 between-subjects conditions?

That models that were identical to each other to the extent that their tasks were identical to each other were able to match all human error rates from the Star Trek tasks says something significant both about the nature of human action selection and modeling of human error. The ACT-R model's inheritance of hierarchical goal representations from the Contention Scheduling Model of Cooper and Shallice (2000) contributed much to its ability to capture a wide range of behaviors in the Star Trek tasks. Productions functioned much like the schemas of the CSM account. The productions responded most often to representations about the internal state of the model, rather than representations of the state of the external worlds, within both the task at the global level and within the individual step at the local level.

Schemas matching to internal task representations at more than one level of behavior created a hierarchical goal representation. This hierarchy of goals was important to model functioning because it enabled behavior that was both efficient for routine tasks and adaptable when situations changed. The adapted behavior was fairly robust, enabling performance that selected the correct action more often than not.

Reflecting this multi-level reality of Star Trek task completion, the model implemented an explicit, hierarchical goal structure meant to keep track of its progress through the task at both levels. Together the hierarchical goal representations held in the goal and imaginal chunks and the pattern-matching, schema-like productions formed the basis for the model's structure and behaviors. Consequently, the model was able to match not just the correct behavior, but the errors generated in the right proportions in 14 separate trial types, including conditions with changing procedures and confusing tasks that shared steps and interfaces, and it was able to recover from errors to finish trials. Clearly the model has captured something important not only to the representation of routine behaviors, but also error in routine procedural behavior, recovery from those errors, and adaptation of routine behaviors to changing circumstances.

The cognitive mechanisms that produced correct performance in all of the trial types, practiced, automatic procedural memory selecting the action to perform and then requesting retrieval of declarative knowledge describing parameters of the action, were flawed. The nature of the flaws produced error rates that matched humans'. The flaws lied in the performance of the declarative memory system. The declarative memory system has a difficult task to perform for us, which is essentially to store all facts that we come across and at some unknown future time reproduce those facts on-demand.

Our minds simply cannot contain unlimited storage and so the declarative system makes the problem tractable by performing a triage on facts (Anderson, 2007, chap. 3). Those facts that are associated to more other facts and that get used relatively more often stay. Those facts that do not have associations with many other facts and that do not get used often go. And often facts that are similar to each other can sometimes be

interchanged. These features of the human declarative memory sometimes become hinderances to human performance when a needed fact is triaged out of our memory and becomes inaccessible just when we need it for performing a step in a task.

Setting certain chunk parameters to just the right values tuned the misretrieval probabilities so that the model would generate error in the right proportions. Ideally these chunk parameters that were fit to the data should reflect some micro-phenomena occurring with those particular steps, such as the confusability of a postcompletion step with its subsequent step as Chung and Byrne (2008) had found.

Meanwhile, the high strength of association between step names that triggered retrieval of change instructions ensured a high likelihood of retrieval of the right change instruction chunk, contributing to total error in appropriate proportions. The steps affected by procedure change, and especially by procedure change and susceptibility to mode errors, did not elicit catastrophic failure of subjects to perform them. Quite the contrary, subjects were able to perform these steps under very difficult circumstances the majority of the times they were presented with them. The combination of productions that requested the retrieval of the change instruction chunks and acted on them and the strength of association between the chunk encoding the flagged step and the chunk encoding the procedure change instruction enabled the model to perform the procedure change tasks and operate the dual-mode Transporter and Jammer task interface with nearly the same proportions of errors generated as humans.

The change instruction procedure enabled the model to adapt an extant routine procedural task representation to an instructed procedure change with only limited increase in error proportion. The model's procedure assumed that subjects entered the

experiment testing session with the procedural knowledge necessary to hold a reference to a future step in working memory, to compare it to a goal representation, and then to use that reference to retrieve a declarative memory representation encoding an instruction indicating a different step to perform instead and then to do that new step.

This is not so far-fetched an assumption. Things happen all the time in daily life that disrupt the routine procedures we engage in. The telephone may ring just as you are about to pour a spoonful of sugar into your coffee. People cope. The model used just two productions plus a small amount of declarative knowledge to cope with procedural change and it did so with similar error proportion consequences as subjects. It therefore seems likely that humans may adapt to changing or otherwise difficult circumstances with simple, pre-packaged sub-routines that attempt to match task states to declarative knowledge. This is a very important feature of the ACT-R model because it allowed the model to match human error rates across the wide range of conditions and trial types present in the behavioral studies. Furthermore this is a very important feature of human procedural behavior because it allowed subjects in the behavioral studies to cope with the wide range of conditions and trial types present in the behavioral studies, including the procedure changes. Even with procedure changes, mean error rates for subjects never exceeded 0.13 for any trial type.

3.5.2. Comparison with the SRN Model

Botvinick and Plaut's (2004) SRN model, on its own, provides a parsimonious explanation of the very narrowly-defined behavior it is asked to explain – but its fault lies in its very narrowly-defined scope to begin with. It cannot even provide a complete account of human action in routine procedures because it does not account for error

correcting behavior. Nor does it generate meaningful quantitative measures of error in a wide variety of contexts. The ACT-R model had only two isomorphs of one task, but it was able to handle five distinct versions of those (Phaser non-intervening, Phaser intervening, Transporter & same-Jammer, different-scanner Jammer, rearranged subtasks Jammer) effectively and can easily be expanded to handle many different and non-isomorphic tasks using the same approaches used here.

The SRN could never have adapted to novel changing circumstances because it only outputs behavioral patterns that it has acquired in training. One issue is the holistic nature of its contextual representation. Because it can only use one, indivisible representation of its context, it cannot form novel combinations of context representation at one level, such as task, with context representation at another level, such as step. Because CSM and ACT-R use separate, discrete representations for each level of behavior, they can perform old steps in new task contexts.

Another problem for the SRN type of account of routine procedural behavior is the postcompletion error observed for the second “Electrical” step of the charging subtask of the Phaser tasks. While working memory effects on error might be explainable by the SRN in terms of contextual representation degradation in general, the postcompletion effect particular to that step would not be explainable by such a mechanism. This is because the postcompletion effect depends on two discrete action representations being similar to each other and the SRN lacks discrete action representations.

4. GENERAL DISCUSSION

It was important that each ACT-R model handle all the same between-subjects conditions as each of the subjects because it was only by constraining the models in that

way that they were able to address effectively the issues of task representation, error generation, error recovery, and procedure change. The results of the behavioral and modeling studies showed that hierarchical goal information does matter for cognitive control across contexts within a task. Also, perceptual-motor considerations weigh on action selection, as the Soft Constraints Hypothesis (Gray, Sims, Fu, & Schoelles, 2006) predicts and as we saw in Experiment 2 with regard to subtask order execution in the Transporter and the Jammer.

The interface displays used in the studies are fairly typical of GUIs in that they feature clusters of interactive buttons of a variety of styles (e.g., checkboxes and radio buttons). Previous work with the Star Trek paradigm (e.g., Chung & Byrne, 2008; Byrne & Bovair, 1997; Byrne, Maurier, Fick, & Chung, 2004) has explored various manipulations of the interface (see Table 1) and has found similar patterns of error and response slow-down. Therefore it is likely that the results observed are not idiosyncratic to the particular task environment used in Experiments 1 and 2.

Most of the effects observed in the behavioral data seemed to have roots in memory mechanisms, such as might be explained by spreading activation combined with high similarity between certain steps. The ACT-R model, using memory structures inspired by CSM, generated per-condition trial type error rates similar to subjects' by just such mechanisms. The model also suggested methods for dealing with tasks that have shared or even duplicated procedures, procedure changes, and recovery from errors.

It is important that the representations supporting human behavior remain flexible enough to adapt to changing circumstances. Having multiple, discrete, hierarchically-organized goal and context representations enables behavior that is both routine – in the

sense of being practiced, skilled and to some degree automatic – and flexible to some degree by placing contextual representations into several pieces that are each coherent and discrete.

A connectionist approach like the SRN, with its holistic task context representation, cannot represent its place within its task independently from representing an arising need to activate error correcting behaviors on the occasion of an error commission. The SRN can only output behaviors that it has learned to associate with the input states to which it has been exposed during its training. So it would have to learn error correction behaviors separably for every step in the task. It cannot generalize behaviors and thus it cannot produce old behaviors in new contexts. Similarly connectionist approaches cannot call upon previously-known mini-procedures to harvest new knowledge and apply them to transform old procedures into new procedures that adapt to a changing task environment.

Though isolated cognitive theories may make some contribution to the body of knowledge, theories that integrate across a wider range of behaviors and conditions are likely to make better contributions to our advancement of human behavior and performance. In the real world, routine procedural memory, working memory, visual perception, and motor systems – to name just a few examples – all work in concert within the context of at least the given task and artifact. By constraining theories to account for a wide range of phenomena we can speak to how those cognitive subsystems interact to give the full range of human behavior. As mentioned in the ACT-R model's discussion section, the SRN model fails to provide accounts of postcompletion error generation, error recovery behavior, and adaptation to procedure change. I opted not to build the

connectionist model that I had originally proposed on the basis of the SRN's inability to address these important phenomena.

The work performed in the behavioral and modeling studies is important for what it says about the nature of human task representation and predicting human error. Human task performance seems to rely on discrete goal representation at more than one level. One goal represents micro-level activities like retrieving action representations and making movements while another goal level represents progress through steps of the task space. Error occurs regularly, mostly as a consequence of recalling the wrong action representation. Fortunately, people have a few pre-packaged strategies that they can rely upon to recover from the errors that they commit. Finally, the multi-level discrete goal representations allow for efficient and flexible action execution that can adapt to changing circumstances.

The ACT-R model that addresses those phenomena is a first step toward developing cognitive engineering tools that can predict not only human error probabilities but also the behavior of humans immediately post-error as well as in reaction to changing circumstances. Cognitive engineering models that can account for these phenomena can be used to evaluate new procedures and tools for high-performance, high-risk domains like aviation, air traffic control, and medicine well before they are put into operation. That way the models can find procedure and tool designs and implementations that pose high risk for inducing error in human operator performance. Cognitive engineering models lacking accounts of task representation structure, error generation, error recovery, and adaptation to change will not be able to perform these important design evaluation tasks for us.

The results gleaned from the human and modeling studies have important practical implications for designers of human-machine interfaces. Human subjects produced the lowest error rates in the Transporter task, the same-scanner Jammer and the reordered subtask Jammer while error rates were relatively high in all versions of the Phaser. Closer analysis revealed that the Phaser's relatively high error rate was due to the presence of two postcompletion steps in that procedure. The model replicated Chung and Byrne's (2008) postcompletion error mechanism wherein working memory constraints on spreading activation to declarative memory coupled with similarity between the postcompletion action representation and the representation of the subsequent action increased probability of error commission at the postcompletion step. Interface and task designers therefore should avoid the inclusion of postcompletion steps (or any other closely-related neighboring steps) in their procedures. Should that be unavoidable, designers should make minimization of demand on working memory at that point in the procedure their top priority.

All error commission by the models were caused by misretrieval of action representations. Misretrievals stemmed from the failure of sufficient activation to spread from references to the correct action representation in the models' goal chunk to the correct action representation chunk in declarative memory. More source activation would have been available to propagate to declarative memory had it not been divided over so many task goals, including representations of current letters for the letter recall task. Here the goal chunk was analogous to human working memory. The models' explanation for error in the Star Trek tasks suggests that working memory should be viewed as one of the

most scarce resources available to the operator. As such, a relatively very high premium should be placed on demand for its use in task environments.

4.1. Unresolved Issues

Though the ACT-R models made an important advance in our understanding of human procedural representation, they were a first attempt at this domain. Refinements remain to be made in order to explain all of the behaviors observed in the two experiments.

First, the model committed no distractor object errors such as was observed in human data during long chains of errors on some single steps. However, this issue could be fairly easily remedied by instantiating a loop that looks for a button and clicks it without having first tried to request the retrieval of a chunk encoding that button.

Another issue regards shared task representations. Why did the structure of all Transporter interface tasks derive from the Transporter task? That is, why was the Jammer task represented in terms of departures from the Transporter task and not vice versa? It is more cognitive miserly to think of two tasks in terms of one task and how the other is different. Probably it could just as easily have been the Transporter task represented in terms of departures from the Jammer.

There is room for improvement in control of retrieval for task-appropriate action representations. The model used two methods to prevent the retrieval of inappropriate action representation chunks: interface slots in the action representation chunks and in the imaginal chunk and large chunk dissimilarity between chunks appropriate to different interfaces. Having two methods to prevent inappropriate chunk retrieval is redundant. Probably one of those methods would have sufficed. Future work might examine whether

both methods are really necessary or whether it might be more parsimonious to use one or the other.

The ACT-R model examined error at a very shallow level compared to its examination in the human data. Since error at the trial type level is composed of error at each of the steps, it would have been preferable to tune the model's performance down to the step level. Analysis of the step completion time data also would be helpful for providing some explanation for the human effects of step time completion. Finally, many chunk parameters were set empirically. It would have been better if they had instead been set a priori based on theoretical motivations.

These things were not made first priority for the modeling effort for two reasons. First, establishing one goal, to match error rates of the 14 per-condition trial types, made the modeling problem tractable. Secondly, by starting with the high-level, quantitative characterization of human performance and error in the experiments, a common description for human representation of routine procedures could be made across all trial types in all four conditions of the two experiments.

Another issue of project scope involved the notion of error in the Star Trek task paradigm. It is important to point out that the Star Trek task environment is a very simplistic one in terms of error commission and error feedback. The most complicated deviation from correct procedure possible is the clicking of a single button since the experiment simply does not allow further action until the correct action is performed. Seldom do people have such a check on the growth of error severity or get immediate feedback informing them that they have committed an error in the task that they were trying to perform. This is a real limitation of the task paradigm. However despite that

limitation subjects still made a significant number of errors and in meaningful patterns, such as the postcompletion errors in the Phaser task. Even more complex error recovery scenarios like real fault diagnosis are important areas, but they lie beyond the scope of the current effort.

4.2. Future Work

The most obvious goal for a future modeling effort would be to extend the current project's model down to the same level of analysis used for the behavioral data – down to the per-step error frequencies and step completion times measured in the behavioral studies. Then the model could provide both the wide-ranging account of the current project's model and the detailed per-step explanation of Chung and Byrne's (2008) model.

Not very many global parameters were fitted, and that is good because using non-default parameter values often bring with them assumptions about cognitive mechanisms. But many chunk parameters were fitted and these parameters were important to model fit. This may present an obstacle to being able to generalize the model to other tasks because of the number of chunk parameters that were empirically fit to the behavioral data after the fact.

On the other hand, the psychological phenomena embodied in many of those parameters would have originated from the training session and the subsequent week intervening between it and the test session. A better model should be able to eliminate fitting those parameters by also modeling the training session and the intervening week.

Hopefully a refined model could generalize to other tasks to explain error generation, error recovery, and coping with the not-so-routine. Although Card, Moran, and Newell

(1983) discussed how to extend GOMS to model some erroneous behaviors and simple error correction behavior, they did not discuss why the errors arose in the first place. Hopefully, eventually, findings from this model about the causes of human error in routine procedures can feedback into cognitive engineering models like GOMS to say something useful about human error probabilities given a task, an interface, and the pre-existing user knowledge brought to bear on the task. Then such engineering models can be put to use designing safer and more economical human-machine systems and procedures.

Finally, future modeling efforts should explore fit metrics that are not dependent on the number of subjects run in each condition. Such a future model might be intended to fall within the average intra-individual deviation, thus passing a sort of Turing test for model fit.

5. REFERENCES

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebière, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036-1060.

Anderson, J. R., Reder, L. M., & Lebiere, C. (1996). Working memory: Activation limitations on retrieval. *Cognitive Psychology, 30,* 221 – 256.

Anderson, J. R., Taatgen, N. A., & Byrne, M. D. (2005). Learning to achieve perfect time sharing: Architectural implications of Hazeltine, Teague, and Ivry (2002). *Journal of Experimental Psychology: Human Perception and Performance, 31(4)*, 749-761.

Botvinick, M., & Plaut, D. C. (2004). Doing without schema hierarchies: A recurrent connectionist approach to normal and impaired routine sequential action. *Psychological Review*, 111, 395 – 429.

Botvinick, M. M., & Bylsma, L. M. (2005). Distraction and action slips in an everyday task: Evidence for a dynamic representation of task context. *Psychonomic Bulletin & Review, 12 (6),* 1011 – 1017.

Botvinick, M. M., & Plaut, D. C. (2006a). Such stuff as habits are made on: A reply to Cooper and Shallice (2006). *Psychological Review, 113 (6)*, 917 – 928.

Byrne, M. D. (2003). A mechanism-based framework for predicting routine procedural errors. In R. Alterman & D. Kirsh (Eds.) *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science. 21(1)*, 31-61.

Byrne, M. D., Maurier, D., Fick, C. S., & Chung, P. H. (2004). Routine procedural isomorphs and cognitive control structures. In C. D. Schunn, M. C. Lovett, C. Lebiere & P. Munro (Eds.), *Proceedings of the Sixth International Conference on Cognitive Modeling* (pp. 52-57).

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

Chung, P. H. (2006). *Changing the interface with minimal disruption: The roles of layout and labels*. Doctoral dissertation, Rice University, Houston, TX.

Chung, P. H., & Byrne, M. D. (2008). Cue effectiveness in mitigating postcompletion errors in a routine procedural task. *International Journal of Human-Computer Studies, 66, 217-232*.

Cooper, R. P., & Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology*, *17*, 297–338.

Cooper, R. P., & Shallice, T. (2006a). Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review*, *113*, 887–916.

Cooper, R. P., & Shallice, T.(2006b) Structured representations in the control of behavior cannot be so easily dismissed: A reply to Botvinick and Plaut (2006). *Psychological Review*, *113*, 929 – 931.

Degani, A., Shafto, M., Kirlik, A. (1999). Modes in human-machine systems: Constructs, Representation, and classification. *The International Journal of Aviation Psychology*, 9, 125-138.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179 –211.

Fennell, K., Sherry, L., Roberts, R. J., & Feary, M. (2006). Difficult access: the impact of recall steps on flight management system errors. *International Journal of Aviation Psychology*, 16(2), 175-196.

Fu, W. T., & Gray, W. D. (2004). Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. *Cognitive Science*, *28*, 901–935.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied, 6 (4)*, 322 – 335.

Gray, W. D., & Fu, W. T. (2004). Soft constraints in interactive behavior: The case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. *Cognitive Science, 28*, 359 – 382.

Gray, W. D., Sims, C. R., Fu, W. T., Schoelles, M. J. (2006). The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review, 113 (3),* 461 – 482.

Howell, D. C. (2002). *Statistical methods for psychology*. Pacific Grove, CA, USA: Duxbury.

John, B. E. (2003). Information processing and skilled behavior. In J. M. Carroll (Ed.) *HCI models, theories, and frameworks: Toward a multidisciplinary science*. San Francisco: Morgan-Kaufmann (pp. 55 – 101).

Kieras, D. E. (1999). *A guide to GOMS model usability evaluation using GOMSL and GLEAN3* (Technical Report). Ann Arbor: University of Michigan.

Landauer, T. K., and Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104,* 211 – 240.

Landauer, T. K., Foltz, P., and Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes, 25,* 259 – 284.

Lashley, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior: The Hixon symposium* (pp. 112–146). New York: Wiley.

Lebiere, C. & Anderson, J. R. (1993). A Connectionist Implementation of the ACT-R Production System. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pp. 635-640.

Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. Davidson, G. Schwartz, & D. Shapiro (Eds.), *Consciousness and self regulation: Advances in research and theory* (Vol. 4, pp. 1–18). New York: Plenum Press.

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.

Rasmussen, J., & Jensen, A. (1974). Mental procedures in real-life tasks: A case study of electronic trouble shooting. *Ergonomics, 17 (3)*, 293 – 307.

Reason, J. (1990). *Human error*. Cambridge, UK: Cambridge University Press.

Reason, J. T., & Mycielska, K. (1982). *Absentminded? The psychology of mental lapses and everyday errors*. Englewood Cliffs, NJ: Prentice-Hall.

Schwartz, M.F., Montgomery, M.W., Buxbaum, L.J., Less, S.S., Carew, T.G., Coslett, H.B., Ferraro, M., Fitzpatrick-De Salme, E.J., Hart, T., & Mayer, N.H. (1998). Naturalistic action impairment in Closed Head Injury. *Neuropsychology, 12(1)*, 13 – 28.

Schwartz, M.F., Montgomery, M.W., Fitzpatrick-De Salme, E.J., Ochipa, C., Coslett, H.B., & Mayer, N.H. (1995). Analysis of a disorder of everyday action. *Cognitive Neuropsychology, 12(8)*, 863 – 892.

Schwartz, M.F., Reed, E.S., Montgomery, M.W., Palmer, C., & Mayer, N.H. (1991). The quantitative description of action disorganisation after brain damage: A case study. *Cognitive Neuropsychology, 8*(5), 381–414.

Taatgen, N. A. (2005). Modeling parallelization and flexibility improvements in skill acquisition: From dual tasks to complex dynamic skills. *Cognitive Science, 29(3)*, 421-455.

Taatgen, N. A., Huss, D., & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General, 137(3)*, 548-565.

Wood, S. D. (2000). *Extending GOMS to human error and applying it to error-tolerant design*. (Doctoral Dissertation, University of Michigan, 2000). UMI Microform No. 9991016.

Wood, S. D., & Kieras, D. E. (2002). Modeling human error for experimentation, training, and error-tolerant design. *Proceedings of The Interservice/Industry Training, Simulation & Education Conference*.

6. APPENDIX A: LSA COSINES FOR PHASER CONTROL LABELS

Table A1
Control Label LSA Cosines for the Semantically-Related Conditions, Within-Subtasks

	Subtask 1		
	"Generator"	"Electrical"	"Kilowatts"
"Generator"		0.69	0.3
"Electrical"	0.69		0.21
"Kilowatts"	0.3	0.21	
Mean LSA Cosine	0.4	0.45	0.26
	Subtask 2		
	"Focus"		
"Lens"	0.71		
	Subtask 3		
	"Shot"		
"Cannon"	0.47		

Table A2
Control Label LSA Cosines for the Semantically-Related Conditions, Between-Subtasks

	"Generator Electrical Kilowatts"	"Lens Focus"	"Cannon Shot"	"Main"
"Generator Electrical Kilowatts"		-0.02	-0.03	0.03
"Lens Focus"	-0.02		0.02	-0.02
"Cannon Shot"	-0.03	0.02		0.03
"Main"	0.03	-0.02	0.03	
Mean LSA Cosine	-0.01	-0.01	0.01	0.01

Table A3
Control Label LSA Cosines for the Semantic Control Condition, Within-Subtasks

	Subtask 1		
	"Smokey"	"Entomologist"	"Filmmaker"
"Smokey"		0.05	-0.01
"Entomologist"	0.05		0.07
"Filmmaker"	-0.01	0.07	
Mean LSA Cosine	0.02	0.06	0.03
	Subtask 2		
	"Reflecting"		
"Solenoid"	-0.06		
	Subtask 3		
	"Headings"		
"Redcoats"	0.01		

Table A4
Control Label LSA Cosines for the Semantic Control Condition, Between-Subtasks.

	"Entomologist Smokey Filmmaker"	"Solenoid Reflecting"	"Redcoats Headings"	"Drafty"
"Entomologist Smokey Filmmaker"		0.01	0.01	-0.02
"Solenoid Reflecting"	0.01		0.02	0.01
"Redcoats Headings"	0.01	0.02		0.02
"Drafty"	-0.02	0.01	0.02	
Mean LSA Cosine	0	0.01	0.02	0

7. APPENDIX B: SUBJECT INSTRUCTIONS

7.1. Aural Instructions, Day 1

Welcome to Star Fleet Academy. Today you will train on several procedures, and before you leave today we will set an appointment for you to return next week and test on those procedures. Main Control is on-screen, and it will direct your training today. On your left is a set of manuals. There is a manual for Main Control, as well as another manual that tells you how you will earn points during the testing session next week and what the cash prizes are for the top three performers in your training group. Please read both manuals before you begin. There is also a manual for each task you will train on. As you come to each task for the first time, please read its manual in its entirety. Then keep the manual as reference while you attempt to do a trial of the task. After you've completed one trial without committing any errors, the computer will ask you to return that manual to me. At that time, please do so and then continue training as Main Control instructs you. Do you have any questions?

7.2. Aural Instructions, Day 2

Today we will test you on the procedures you learned last week, but in addition there will be a letter recall task for you to perform while you are performing those other tasks. Doing both at the same time is fairly difficult, but please just do the best that you can. Go as fast as you can while committing as few errors as you can. At the end there will be a brief questionnaire.

7.3. Written Instructions, Day 2

PLEASE READ THESE INSTRUCTIONS CAREFULLY

In order to ensure that our pilots are able to operate all systems in any field situation, we will be testing your on-the-fly thinking and ability to adapt. Being able to operate Starfleet systems under any external circumstance is imperative, particularly in emergency situations. Some of the interfaces you will be using may change halfway through the examination, and the system will warn you of this change. Please do your best to continue with the tasks and complete them as you did previous to the change.

Additionally, your ability to monitor and recall information will be tested. Please be sure to wear the headphones during the experiment. While you are doing the tasks that you trained on during the last session, please pay attention to the alphabet letters recited. Be ready at any time to recall the last three letters that you have heard, in the order in which you heard them. A “recall” window will pop up when it is time for you to recall those three letters. A warning tone will be played to notify you of an incorrect recall.

After completing the experiment you will be asked to complete a brief questionnaire.

7.4. Written Instructions, Day 2, Change Onset Instructions

As explained previously, system controls in the next task will now be changed in the following trials to simulate an emergency situation in which console damage has been sustained. Please do your best to complete the tasks as before.

Your next task is the Phaser. Now, instead of clicking “Electrical”, “Generator”, “Kilowatts”, and “Electrical” as before, you will now click “Electrical”, “Generator”, then “Lens”, the focus slider, and “Lens”. Then battery Generator meter will begin to fill, indicating that the battery is charging. As before, wait for the meter to reach an appropriate level, then click “Kilowatts” and finally “Electrical”.

8. APPENDIX C: ADDITIONAL METHODS DETAIL FOR THE BEHAVIORAL STUDY

8.1. Main Control

The Main Control interface of the experiment functioned as a coordinator of training and of testing. During the training phase of the experiment, Main Control displayed a message indicating to the subject which task to train on. When subjects erred in the procedures they were training on, the task window returned them to Main Control. At this point Main Control displayed a message indicating which step in the procedure should have been performed. For example, if a subject was training on the Phaser task and clicked “lens” when “focus” should have been clicked, the Phaser task would abort, Main Control would come back on-screen, and Main Control would display the message, “Click the ‘focus’ button.” When subjects correctly finished a trial, Main Control displayed how many trials they had completed successfully and what task to train on for the next trial. Figure 22 shows Main Control during training.

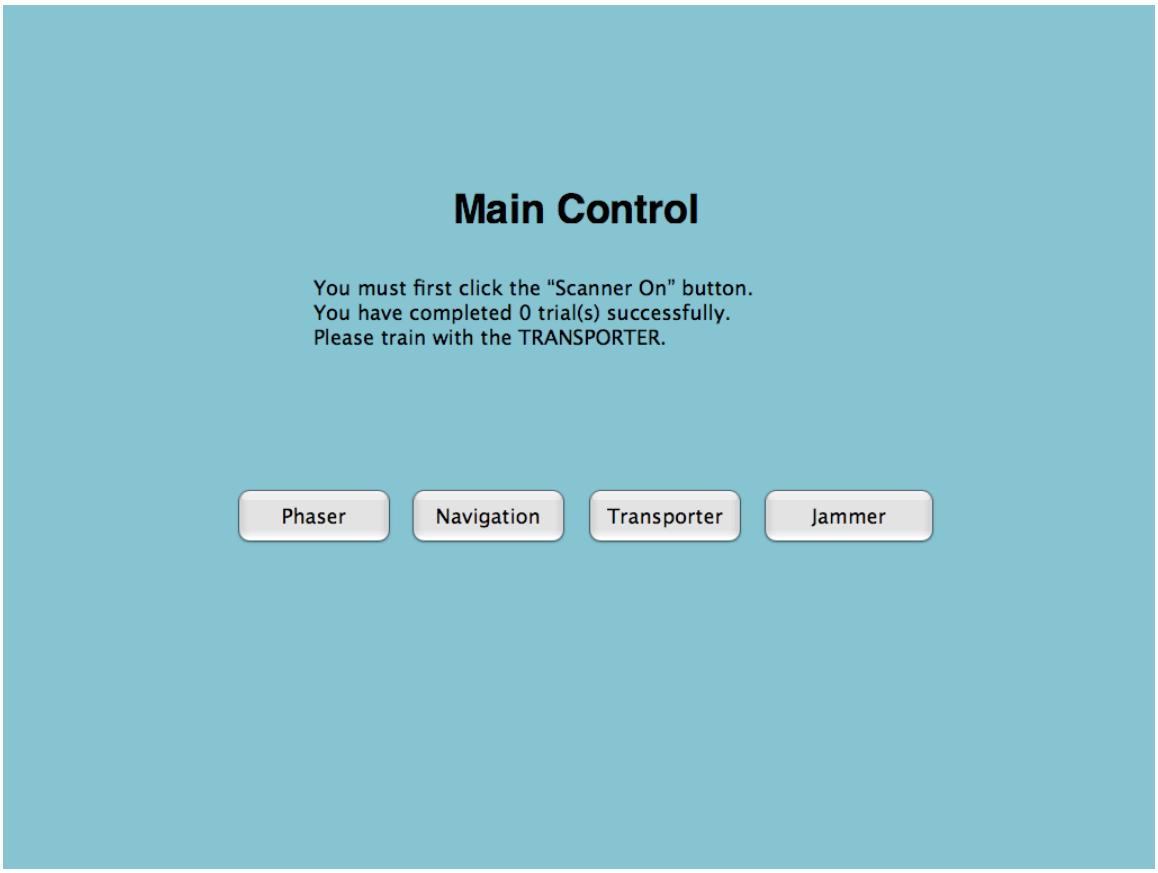

Figure 22. Main Control, training.

During the testing session, Main Control again indicated which task to perform for each trial. At the conclusion of each trial, Main Control indicated how many seconds it took to complete that trial and how many times the subject had erred. During the testing session Main Control never displayed messages indicating which action should have been performed at a given step in a task as it did during the training session. Additionally, if the subject completed the trial fast enough to earn bonus points, Main Control displayed a message stating such. See the Testing Session subsection for an explanation of the rationale and mechanics of the scoring system. Figure 23 shows Main Control during testing.

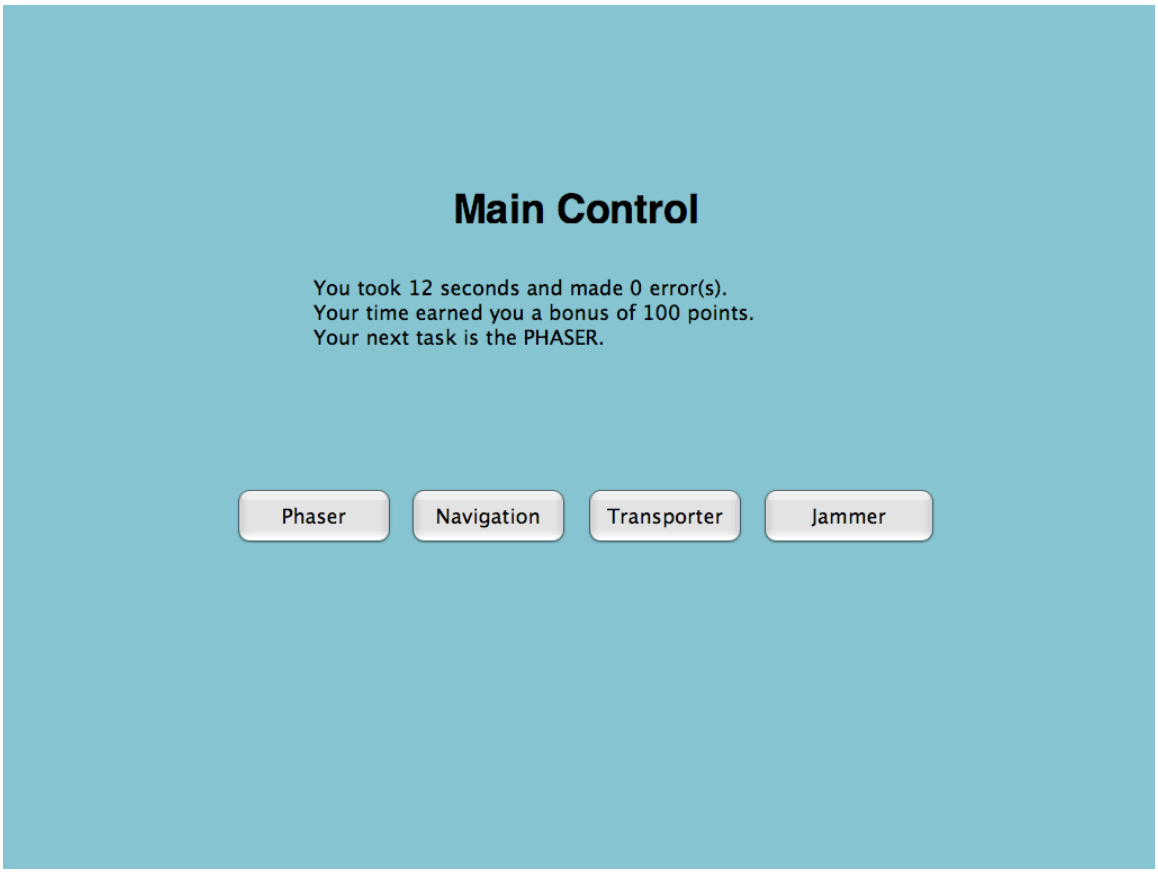

Figure 23. Main Control, testing.

8.2. Additional Phaser Detail

For all Phaser versions, there was some chance that the experiment would require the subject to repeat much of the trial. After subjects shot the Phaser, the experiment software displayed an outcome message in the status box on the lower-left of the screen, “Romulan vessel destroyed,” “Romulan vessel hit but not destroyed,” or “Phaser missed Romulan vessel.” In the first case, subjects had only to click “Shot” and “Main” to finish the Phaser task. The latter two cases were equivalent and meant that subjects had to perform the Phaser task all over again. This resulted in another trial’s worth of data, but it did not count toward trial count for the sake of procedure change in the untrained intervening subtask condition.

The Phaser hit or miss was an independent event governed by the following rules presented in decreasing order of priority. There was a 1% chance of scoring a hit regardless of the distance from the crosshairs to the target. Firing the Phaser when the target was more than 70 pixels from the crosshairs resulted in a miss. Otherwise the Phaser hit the target when a randomly chosen integer from the set {0, 99} was less than or equal to the lesser of 85 or 95 minus the crosshairs-to-target distance times one plus the focus value of the Phaser divided by 20, where the focus value of the Phaser was the distance from the nearest point on the left edge of the interface to the position clicked on the focus slider.

8.3. The Navigation Task

The Star Trek set of tasks uses a "Navigation" task as a filler. The Navigation task is a simple arithmetic task wherein the subjects were to compare a "Programmed Heading" to a "Current Heading" and compute the difference for the X-, Y-, and Z-coordinates. Subjects first clicked the "Confirm Course" button, then subtracted the current heading from the programmed heading, entering the difference for each coordinate into the three "Course Correction" fields. A three-axis graph plotted the programmed and current headings mostly to add an engaging, interactive component to the task. However, when the differences between both sets of X, Y, and Z values were zero a quick glance at the graph, showing overlapping headings, could tell the user that the courses are the same faster than subjects could read all six values and make three comparisons. Figure 24 displays the Navigation task's interface.

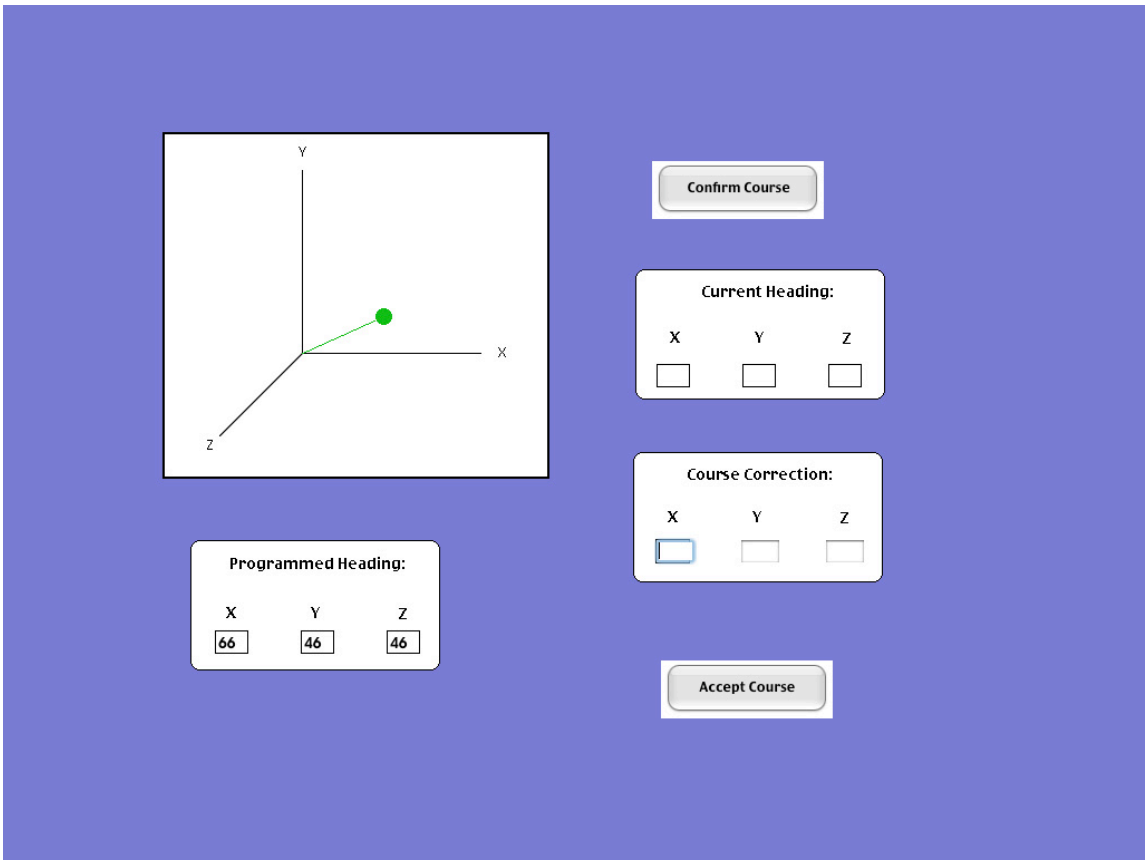

Figure 24. The Navigation interface.

8.4 Additional Transporter Task Detail

Transporter and Jammer trials re-started for the same reasons and with the same consequences as the Phaser. That is, re-starting the trial resulted in more data being collected from subjects and subjects had to start the Transporter task over again after clicking the target and then clicking “Synchronous Mode”. Upon clicking “Synchronous Mode” after having clicked the Transporter target, the Transporter displayed one of these three status messages: “Beam successful--return to main control,” “Beam failed--jammed by hostile signal,” or “Beam failed--beam too weak.” The last two messages both indicated a miss and that it would be necessary to repeat the trial. For the conditions in which subjects were free to choose which order to perform the frequency and

synchronization subtasks, they were also free to choose which order to perform these subtasks when the trial repeated.

The Transporter determined hits and misses according to the following rules, in order of decreasing priority. If the subject had clicked “Lock Signal” before all but one of the scanning dots had disappeared from the scanner display, then the Transporter missed. If the absolute value of the difference between the number entered for the frequency minus a randomly chosen integer from the set {0, 99} was less than 41, then the Transporter hit. If that difference was less than 90, then the Transporter missed. If that difference was 90 or greater, then the probability of a hit was 0.5.

9. APPENDIX D: EXPERIMENTER SCRIPT

X84 Experimenter Script

Starting the Day

- Login to the eMacs
- Load X84.lisp
- Get out the subject registry and folders for consent and debriefing forms.

Subject Arrival

Greet and ask whether today is their first day or second day of participation.

Day 1

- Credit or pay?
- Give two copies of appropriate consent form:
  - "Please read it over and ask me any questions you may have. If you agree to participate, please fill out one copy, give that to me and keep the other for your records. Then please fill out the next available line on this form"
  - Indicate the X84 registration form
- Retrieve the appropriate manual set, place it next to a station's keyboard.
- Begin the experiment program: type "(begin-experiment)" and hit return.
- Give instructions:
  - "Welcome to Star Fleet Academy. Today you will train on several procedures, and before you leave today we will set an appointment for you to return next week to test on those procedures. Main Control is on-screen, and it will direct your training today. On your left is a set of manuals. There is a manual for Main Control, as well as another manual that tells you how you will earn points during the testing session next week and what the cash prizes are for the top three performers in your training group. Please read both manuals before you begin. There is also a manual for each task you will train on. As you come to each task for the first time, please read its manual in its entirety. Then keep the manual as reference while you attempt to do a trial of the task. After you've completed one trial without committing any errors, the computer will ask you to return that manual to me. At that time, please do so and then continue training as Main Control instructs you. Do you have any questions? … Please begin when you are ready.
- Let them run, collecting manuals as needed
- Subject finishes:
  - Set appointment for next week
  - If subject participates for pay, pay for day one ($10)
- Upload the day's data files to the server, /Public/X Support/X84 Frank Trek Semantic Space/X84 Data/. Please be sure to put data from each eMac into its appropriate sub-folder.

Day 2

- Name?
  - Find his name on the subject assignment sheet for condition & subject numbers.
- Start the experiment program.
- Give instructions:
  - "Today we will test you on the procedures you learned last week, but in addition there will be a letter recall task for you to perform while you are performing those other tasks. Doing both at the same time is fairly difficult, but please just do the best that you can. Go as fast as you can while committing as few errors as you can. At the end there will be a brief questionnaire. When you are ready to begin, please read the instructions on-screen and begin."
- Subject finishes:
  - Thank for participation, give debriefing sheet.
  - If subject participates for pay, pay for day two ($15)

Finishing up the Day

- Put away all paper work
- Count the cash left, contact Frank if there is $50 or less
- Copy all data from today to the appropriate folder on the server
- Logout from eMacs

Other Things to Know

- Be prepared for an experiment crash: Use command-. on the error window or command-w on the errant task window to get back to Main Control. Then have the subject click the button for the task he was performing when the experiment crashed. Once the task resumes, X84 should be fine. If not, try to contact Frank (http://chil.rice.edu/labonly/contact.html has lab members' contact information). If no luck, then apologize and reschedule. Try to copy the text of the error message and get that to Frank, as well as any information you can gather about what was going on at the time of the crash.
- If the subject asks a question to which you don't know the answer, and Frank's not around, tell him he can email Frank or Dr. Byrne (contact information on Experimetrix) or the debriefing form, but only give him the debriefing form if he's finished day 2.

10. APPENDIX E: ADDITIONAL DETAIL REGARDING SHARED TASK REPRESENTATIONS AND HANDLING OF TASK PROCEDURE CHANGE

The same mechanism handled the shared representation of the Transporter and Jammer tasks as well as procedure change in the Jammer and Phaser tasks. For the static different-Jammer, the model had three productions specific to the Jammer and five "change instruction" chunks, named for the role that they played in handling procedure change. These change instruction chunks specified a Transporter step to be interrupted, the Jammer step to perform instead, and the next step of the basic Transporter procedure to be interrupted. Since the different-Jammer task was identical to the Transporter task but with some steps rearranged, the goal chunk for the different-Jammer flagged a Transporter action that occurred in a different order in the different-Jammer by placing another copy of a reference to that action in a third state slot. When the model got to that flagged step at the retrieve phase of the RFMAV loop, as indicated by a coinciding of the "retrieve" local state and an equality of the global state value and the flagged step value, the "flagged-step-Jammer" production matched.

Instead of requesting the retrieval of a action representation chunk, the flagged-step-Jammer production requested the retrieval of a change instruction chunk with a current-flag slot value equal to the value of the flagged step. Once that change instruction chunk had been retrieved, the "got-change instruction-Jammer" production fired. This production set the global state value to the Jammer step to be performed instead and directed the model back to the retrieve phase of the RFMAV loop. Assuming the change instruction retrieval request resulted in the retrieval of the correct change instruction chunk, the model then continued through the RFMAV loop with the correct Jammer step.

That step's corresponding verify production would set the global state slot to the next step appropriate for the Transporter task.

For example, if the model was beginning the different-scanner version of the Jammer task the correct first step would be "active scan," whereas the correct first step of the Transporter task was "scanner on." The flagged-step-Jammer production would match to the condition wherein both the global state and flagged-step slots of the goal chunk are set to "scanner on." The flagged-step-Jammer production then would request the retrieval of the change instruction chunk that had "scanner on" in its flagged-step slot. That chunk would have another slot, do-step, specifying the action that the model should perform instead, "active scan," as well as the identity of the next Transporter step to flag – the Transporter step that would follow "active scan," "lock signal." The model would then continue its way around the RFMAV loop until it got to the retrieval phase with the global state and flagged-step slots of the goal chunk set to "lock signal."

When the model came back around to the retrieve phase with the global state of "lock signal," the global state would be equal to the flagged step indicated by the last retrieved change instruction chunk, and so the flagged-step-Jammer production fired and the model progressed again through its change instruction side loop. Once the model had completed the scanner subtask, it would have advanced to a portion of the Jammer task that was identical to the Transporter task, and so effectively it just performed the Transporter task. Because of the model's reliance on this change instruction mechanism, it performed two retrievals for every Jammer step that is different from the Transporter step, instead of one retrieval per step that is the norm for the other tasks. Retrieving chunks from declarative memory was a noisy process fraught with error, and this was the genesis for the different-

Jammer's relatively high error rate in its scanner subtask. Finally, there was a third change instruction production that handled error recovery in the case of the model committing an error while operating on a change instruction. It will be discussed in the section regarding error recovery.

For the procedure-change tasks, the model operated as above except that it started the experiment run with only the productions and not the declarative chunks encoding the procedure change knowledge. When the procedure change occurred, the experiment software performed an abstracted version of the change instruction presentation – it simply added the change instruction chunks to the model's declarative module. The experiment also changed the goal chunk to reflect the changed context of the task. With those three components in place, the productions, the change instruction chunks, and the procedure-change goal context, the model could then perform the new version of the Jammer or Phaser task.

Table A5
Model Global Parameter Values

Parameter	Value	Reason
egs (expected gain s)	2	empirical fit
mp (mismatch penalty)	1.8	empirical fit
ans (activation noise standard deviation)	0.23	empirical fit
mas (maximum associative strength)	2	empirical fit
rt (retrieval threshold)	-2	ensured that most retrieval requests would return some chunk
ol (optimized learning)	4	used by declarative module to implement a computationally simple form of the base level learning equation
md (maximum difference)	-10	ensured that action representation chunks for the wrong interface would not be retrieved

Table A6
Model Chunk Parameter Values: Base Levels.

Chunk	Value	Reason
second instance of the "Electrical" step	-1.2	empirical fit
second instance of the "Shot" step	-1.1	empirical fit
scanner on	-0.4	empirical fit
scanner off	-0.4	empirical fit
second instance of the "Synchronous Mode" step	-0.4	empirical fit

Table A7
Model Chunk Parameter Values: Strengths of Association.

Condition & Task	Chunk A	Chunk B	Value	Reason
No procedure change, different scanner subtask Jammer	scanner on change instruction	scanner on	1.6	empirical fit
No procedure change, different scanner subtask Jammer	lock signal change instruction	lock signal	1.6	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency change instruction	enter frequency	1.6	empirical fit
No procedure change, different scanner subtask Jammer	scanner off change instruction	scanner off	1.6	empirical fit
No procedure change, different scanner subtask Jammer	active scan change instruction	active scan	1.6	empirical fit
No procedure change, different scanner subtask Jammer	scanner on	scanner on change instruction	1.6	empirical fit
No procedure change, different scanner subtask Jammer	lock signal	lock signal change instruction	1.6	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency	enter frequency change instruction	1.6	empirical fit
No procedure change, different scanner subtask Jammer	scanner off	scanner off change instruction	1.6	empirical fit
No procedure change, different scanner subtask Jammer	active scan	active scan change instruction	1.6	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
No procedure change, different scanner subtask Jammer	scanner on change instruction	lock signal	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on change instruction	enter frequency	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on change instruction	scanner off	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on change instruction	active scan	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal change instruction	scanner on	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal change instruction	enter frequency	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal change instruction	scanner off	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal change instruction	active scan	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency change instruction	scanner on	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency change instruction	lock signal	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency change instruction	scanner off	0.4	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
No procedure change, different scanner subtask Jammer	enter frequency change instruction	active scan	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off change instruction	scanner on	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off change instruction	lock signal	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off change instruction	enter frequency	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off change instruction	active scan	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan change instruction	scanner on	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan change instruction	lock signal	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan change instruction	enter frequency	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan change instruction	scanner off	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal	scanner on change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency	scanner on change instruction	0.4	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
No procedure change, different scanner subtask Jammer	scanner off	scanner on change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan	scanner on change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on	lock signal change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency	lock signal change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off	lock signal change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan	lock signal change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on	enter frequency change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal	enter frequency change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off	enter frequency change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan	enter frequency change instruction	0.4	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
No procedure change, different scanner subtask Jammer	scanner on	scanner off change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal	scanner off change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency	scanner off change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	active scan	scanner off change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner on	active scan change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	lock signal	active scan change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	enter frequency	active scan change instruction	0.4	empirical fit
No procedure change, different scanner subtask Jammer	scanner off	active scan change instruction	0.4	empirical fit
Procedure change Phaser, post-change	kilowatts	kilowatts change instruction	2.5	empirical fit
Procedure change Phaser, post-change	cannon	cannon change instruction	2.5	empirical fit
Procedure change Phaser, post-change	lens	lens change instruction	2.5	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Phaser, post-change	kilowatts change instruction	kilowatts	2.5	empirical fit
Procedure change Phaser, post-change	cannon change instruction	cannon	2.5	empirical fit
Procedure change Phaser, post-change	lens change instruction	lens	2.5	empirical fit
Procedure change Phaser, post-change	kilowatts	cannon change instruction	-0.5	empirical fit
Procedure change Phaser, post-change	kilowatts	lens change instruction	-0.5	empirical fit
Procedure change Phaser, post-change	cannon	kilowatts change instruction	-0.5	empirical fit
Procedure change Phaser, post-change	cannon	lens change instruction	-0.5	empirical fit
Procedure change Phaser, post-change	lens	kilowatts	-0.5	empirical fit
Procedure change Phaser, post-change	lens	cannon	-0.5	empirical fit
Procedure change Phaser, post-change	cannon change instruction	kilowatts	-0.5	empirical fit
Procedure change Phaser, post-change	lens change instruction	kilowatts	-0.5	empirical fit
Procedure change Phaser, post-change	kilowatts change instruction	cannon	-0.5	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Phaser, post-change	lens change instruction	cannon	-0.5	empirical fit
Procedure change Phaser, post-change	kilowatts	lens	-0.5	empirical fit
Procedure change Phaser, post-change	cannon	lens	-0.5	empirical fit
Procedure change Jammer, post-change	scanner on change instruction	scanner on	1	empirical fit
Procedure change Jammer, post-change	lock signal change instruction	lock signal	2	empirical fit
Procedure change Jammer, post-change	enter frequency change instruction	enter frequency	1.3	empirical fit
Procedure change Jammer, post-change	scanner off change instruction	scanner off	2	empirical fit
Procedure change Jammer, post-change	active scan change instruction	active scan	1.6	empirical fit
Procedure change Jammer, post-change	scanner on	scanner on change instruction	1	empirical fit
Procedure change Jammer, post-change	lock signal	lock signal change instruction	2	empirical fit
Procedure change Jammer, post-change	enter frequency	enter frequency change instruction	1.3	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Jammer, post-change	scanner off	scanner off change instruction	2	empirical fit
Procedure change Jammer, post-change	active scan	active scan change instruction	1.6	empirical fit
Procedure change Jammer, post-change	scanner on change instruction	lock signal	0.2	empirical fit
Procedure change Jammer, post-change	scanner on change instruction	enter frequency	0.5	empirical fit
Procedure change Jammer, post-change	scanner on change instruction	scanner off	0.2	empirical fit
Procedure change Jammer, post-change	scanner on change instruction	active scan	0.4	empirical fit
Procedure change Jammer, post-change	lock signal change instruction	scanner on	0.6	empirical fit
Procedure change Jammer, post-change	lock signal change instruction	enter frequency	0.5	empirical fit
Procedure change Jammer, post-change	lock signal change instruction	scanner off	0.2	empirical fit
Procedure change Jammer, post-change	lock signal change instruction	active scan	0.4	empirical fit
Procedure change Jammer, post-change	enter frequency change instruction	scanner on	0.6	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Jammer, post-change	enter frequency change instruction	lock signal	0.2	empirical fit
Procedure change Jammer, post-change	enter frequency change instruction	scanner off	0.2	empirical fit
Procedure change Jammer, post-change	enter frequency change instruction	active scan	0.4	empirical fit
Procedure change Jammer, post-change	scanner off change instruction	scanner on	0.6	empirical fit
Procedure change Jammer, post-change	scanner off change instruction	lock signal	0.2	empirical fit
Procedure change Jammer, post-change	scanner off change instruction	enter frequency	0.5	empirical fit
Procedure change Jammer, post-change	scanner off change instruction	active scan	0.4	empirical fit
Procedure change Jammer, post-change	active scan change instruction	scanner on	0.2	empirical fit
Procedure change Jammer, post-change	active scan change instruction	lock signal	0.2	empirical fit
Procedure change Jammer, post-change	active scan change instruction	enter frequency	0.5	empirical fit
Procedure change Jammer, post-change	active scan change instruction	scanner off	0.2	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Jammer, post-change	lock signal	scanner on change instruction	0.2	empirical fit
Procedure change Jammer, post-change	enter frequency	scanner on change instruction	0.5	empirical fit
Procedure change Jammer, post-change	scanner off	scanner on change instruction	0.2	empirical fit
Procedure change Jammer, post-change	active scan	scanner on change instruction	0.4	empirical fit
Procedure change Jammer, post-change	scanner on	lock signal change instruction	0.6	empirical fit
Procedure change Jammer, post-change	enter frequency	lock signal change instruction	0.5	empirical fit
Procedure change Jammer, post-change	scanner off	lock signal change instruction	0.2	empirical fit
Procedure change Jammer, post-change	active scan	lock signal change instruction	0.4	empirical fit
Procedure change Jammer, post-change	scanner on	enter frequency change instruction	0.6	empirical fit
Procedure change Jammer, post-change	lock signal	enter frequency change instruction	0.2	empirical fit

Condition & Task	Chunk A	Chunk B	Value	Reason
Procedure change Jammer, post-change	scanner off	enter frequency change instruction	0.2	empirical fit
Procedure change Jammer, post-change	active scan	enter frequency change instruction	0.4	empirical fit
Procedure change Jammer, post-change	scanner on	scanner off change instruction	0.6	empirical fit
Procedure change Jammer, post-change	lock signal	scanner off change instruction	0.2	empirical fit
Procedure change Jammer, post-change	enter frequency	scanner off change instruction	0.5	empirical fit
Procedure change Jammer, post-change	active scan	scanner off change instruction	0.4	empirical fit
Procedure change Jammer, post-change	scanner on	active scan change instruction	0.2	empirical fit
Procedure change Jammer, post-change	lock signal	active scan change instruction	0.2	empirical fit
Procedure change Jammer, post-change	enter frequency	active scan change instruction	0.5	empirical fit
Procedure change Jammer, post-change	scanner off	active scan change instruction	0.2	empirical fit

Table A8
Model Chunk Parameter Values: Similarities

Condition & Task	Chunk Pair	Value	Reason
No procedure change, intervening subtask Phaser	second instance of shot, main control	-0.7	empirical fit, simulation of post completion error
No procedure change, intervening subtask Phaser	second instance of electrical, cannon	-0.7	empirical fit, simulation of post completion error
No procedure change, different scanner subtask Jammer	scanner on, active scan	-0.75	empirical fit
No procedure change, different scanner subtask Jammer	scanner on, lock signal	-0.85	empirical fit
No procedure change, different scanner subtask Jammer	scanner on, scanner off	-0.85	empirical fit
No procedure change, different scanner subtask Jammer	active scan, lock signal	-0.9	empirical fit
No procedure change, different scanner subtask Jammer	active scan, scanner off	-0.9	empirical fit
No procedure change, different scanner subtask Jammer	lock signal, scanner off	-0.72	empirical fit
No procedure change, different scanner subtask Jammer	second instance of synchronous mode, main control	-0.9	empirical fit
No procedure change, different scanner subtask Jammer	scanner on, enter frequency	-0.9	empirical fit

Condition & Task	Chunk Pair	Value	Reason
No procedure change, different scanner subtask Jammer	active scan, enter frequency	-0.9	empirical fit
No procedure change, different scanner subtask Jammer	lock signal, enter frequency	-0.9	empirical fit
No procedure change, different scanner subtask Jammer	scanner off, enter frequency	-0.7	empirical fit
Procedure change, intervening subtask Phaser, pre-change	second instance of shot, main control	-0.7	empirical fit, simulation of post completion error
Procedure change, intervening subtask Phaser, pre-change	second instance of electrical, lens	-0.7	empirical fit, simulation of post completion error
Procedure change, intervening subtask Phaser, post-change	second instance of electrical, lens	-1	empirical fit
No procedure change, same scanner subtask Jammer	scanner on, active scan	-0.75	empirical fit
No procedure change, same scanner subtask Jammer	scanner on, lock signal	-0.85	empirical fit
No procedure change, same scanner subtask Jammer	scanner on, scanner off	-0.85	empirical fit
No procedure change, same scanner subtask Jammer	active scan, lock signal	-0.9	empirical fit
No procedure change, same scanner subtask Jammer	active scan, scanner off	-0.9	empirical fit

Condition & Task	Chunk Pair	Value	Reason
No procedure change, same scanner subtask Jammer	lock signal, scanner off	-0.72	empirical fit
No procedure change, same scanner subtask Jammer	Transporter power, first instance of synchronous mode	-0.75	empirical fit
No procedure change, same scanner subtask Jammer	second instance of synchronous mode, main control	-0.9	empirical fit
No procedure change, same scanner subtask Jammer	scanner off, enter frequency	-0.7	empirical fit
No procedure change, no intervening subtask Phaser	second instance of shot, main control	-0.7	empirical fit, simulation of post completion error
No procedure change, no intervening subtask Phaser	second instance of electrical, lens	-0.7	empirical fit, simulation of post completion error
Procedure change Jammer, pre-change	scanner on, active scan	-0.75	empirical fit
Procedure change Jammer, pre-change	scanner on, lock signal	-0.85	empirical fit
Procedure change Jammer, pre-change	scanner on, scanner off	-0.85	empirical fit
Procedure change Jammer, pre-change	active scan, lock signal	-0.9	empirical fit
Procedure change Jammer, pre-change	active scan, scanner off	-0.9	empirical fit
Procedure change Jammer, pre-change	lock signal, scanner off	-0.72	empirical fit
Procedure change Jammer, pre-change	second instance of synchronous mode, main control	-0.9	empirical fit

Condition & Task	Chunk Pair	Value	Reason
Procedure change Jammer, pre-change	scanner on, enter frequency	-0.9	empirical fit
Procedure change Jammer, pre-change	active scan, enter frequency	-0.9	empirical fit
Procedure change Jammer, pre-change	lock signal, enter frequency	-0.9	empirical fit
Procedure change Jammer, pre-change	scanner off, enter frequency	-0.7	empirical fit
Semantic control Phaser	second instance of shot, main control	-0.7	empirical fit, simulation of post completion error
Semantic control Phaser	second instance of electrical, lens	-0.8	empirical fit, simulation of post completion error
Jammer, different subtask order	scanner on, active scan	-0.75	empirical fit
Jammer, different subtask order	scanner on, lock signal	-0.85	empirical fit
Jammer, different subtask order	scanner on, scanner off	-0.85	empirical fit
Jammer, different subtask order	active scan, lock signal	-0.9	empirical fit
Jammer, different subtask order	active scan, scanner off	-0.9	empirical fit
Jammer, different subtask order	lock signal, scanner off	-0.72	empirical fit
Jammer, different subtask order	second instance of synchronous mode, main control	-0.9	empirical fit
Jammer, different subtask order	scanner on, enter frequency	-0.9	empirical fit

Condition & Task	Chunk Pair	Value	Reason
Jammer, different subtask order	active scan, enter frequency	-0.9	empirical fit
Jammer, different subtask order	lock signal, enter frequency	-0.9	empirical fit
Jammer, different subtask order	scanner off, enter frequency	-0.7	empirical fit

Note: Unless otherwise listed in this table, action representation chunks appearing within the same subtask had a similarity value of -0.95 and action representation chunks appearing in different subtasks within the same task had a similarity value of -1. Action representation chunks not appearing in this table or this note defaulted to the maximum different global parameter value, -10. These were action representation chunk pairs that were not applicable to tasks using the same interface, such as "electrical" and "scanner on."