The Pennsylvania State University

The Graduate School

College of Engineering

PROCEDURAL SKILLS:
FROM LEARNING TO FORGETTING

A Dissertation in

Industrial Engineering

by

Jong Wook Kim

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2008

The dissertation of Jong Wook Kim was reviewed and approved* by the following:

*Signatures are on file in the Graduate School.

Abstract

Can we help people forget less by knowing how they learn? Can we decrease forgetting by modifying what they learn? These have been long-standing questions in applied cognitive science and engineering. My dissertation study addresses *the decay of procedural skills*. A study paradigm was created to investigate learning and forgetting of procedural skills in a laboratory setting. Human participants learned and performed a set of novel spreadsheet tasks that are declarative or procedural, and perceptual-motor or cognitive. To examine procedural skills on learning and forgetting, one group of participants used key-based commands and the other group used a novel mouse and menus to complete the task. Participants were able to learn the task well in four learning sessions, confirming the Power Law of learning. Mouse users did not learn or perform better than keyboard users. Retention intervals (6-day, 12-day, or 18-day) showed clear effects on the amount of forgetting. Two modalities (mouse or keyboard), however, did not provide any statistically different rates of forgetting on the first return. When it comes to relearning (2nd and 3rd returns), mouse users showed significantly decreased mean task completion time, indicating relearning occurred in mouse users more effectively than keyboard users. The ACT-R theory, which is used as the main theoretical background, was tested against human data with regard to learning and forgetting. The skill retention model in ACT-R was developed to predict a mouse user's learning and forgetting performance in one subtask. The model predicted the learning performance with $r^2 = 0.8$ and $RMSSD = 1.8$, when compared with human data. The skill retention model proved that an ACT-R model is able to predict learning performance. Human performance modeling using ACT-R can be used to evaluate efficacy of a training regimen by predicting learning performance, making contributions to workforce engineering both in industry and in military.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Discussions with Dr. Richard Koubek opened up my eyes to see engineering issues in training and education at the beginning of my pursuit toward the doctoral degree. I strongly got motivated to solve the issue. My motivation has taken me to delve into multi-disciplinary minds including industrial engineering, cognitive science, psychology, and computer science.

Dr. Frank E. Ritter shaped and tuned my motivation and interests. My interests and his academic disciplines have been synergistically merged. I thank that he was willing to be my co-advisor and helped me intellectually and financially to get my work done. In particular, he had brought me to a world where I met the profound work of Allen Newell and John Anderson. I also would like to express my gratitude to Dr. Andris Freivalds and Dr. David Nembhard. They have given me numerous comments and improved my dissertation.

The project was supported by the Office of Naval Research, which helps me to finish my dissertation and to continue research on learning and forgetting. Also, College of Information Sciences and Technology at the Pennsylvania State University provided initial support and helped me to run human-involved experiments.

I thank members in the Applied Cognitive Science laboratory. Olivier Georgeon who went out for lunch with me and gave useful comments and encouragement for my dissertation. Jonathan Morgan helped me to improve my slides for the final defense and my writing of the dissertation. Jennifer Bittner also answered my questions related to the field of psychology. Mark Cohen gave me useful feedback to my practice talk at the lab. Sangwon Lee helped to run experiments. I also thank the ACT-R community. Particularly, Dan Bothell at Carnegie Mellon University provided useful comments for modeling and simulating the ACT-R model.

There are many people who helped me and encouraged me when I was exhausted by research work: Seunghoon Bang, Oujung Kwon, and more.

Finally, I thank my wife, Jihye, for giving me abundant love and support. Now, I want to enjoy sunny weather with my adorable twelve-month old daughter, Hannah, and my wife, in Happy Valley, State College.

Chapter 1
Introduction

Preparedness and readiness play an important role in coping with unexpected incidents. In 2005, Hurricane Katrina seriously devastated lives and homes in Louisiana and Mississippi. Under this emergency situation, the mission of the emergency responders is to provide rapid and accurate responses against a trail of devastation. Training plays a key role to provide the necessary responses against an incident.

Training is of importance not only in industry but also in the military domain. In particular, lots of research has been conducted to provide effective training paradigms to prepare military personnel in the United States. However, most research has been solely given consideration from the perspective of learning instead of forgetting, even though forgetting is an inevitable human feature

This study demonstrates how multi-disciplinary aspects of industrial engineering, cognitive science, and computer science can be utilized to better shape workforce training and how the phenomenon of forgetting can be incorporated toward a novel improved training paradigm.

1.1 The Role of Workforce Training

The growing concerns about the unexpected and unwanted terrorist attacks inevitably necessitate counter-terrorism training drills in major cities of the United States. The first responders including firefighters, bomb squad personnel, military personnel, or even doctors and nurses, would get trained to guarantee expeditious and effective responses under chaotic terrorist attacks.

The Washington Post reported the results of a five-day counter-terrorism training exercise (Johnson, 2003). The drill, which was the first large-scale training in the United States, was aimed at testing the readiness of the first responders under the simulated explosions of dirty bombs in Seattle and the simulated release of biological agents in Chicago. According to the report, the training exercises cost \$16 million and included more than 8,500 people from federal, state, and local agencies. We might need to budget for this type of training, but the cost of training brings up an important research question that how to optimize training programs. Thus, while reducing frequency of training, can we train first responders to acquire more robust skills that are resistant against forgetting and performance decrements?

First responders mainly acquire knowledge and skills from hands-on training or education. By the way, it is crucial to note that the acquired knowledge and skills would be elicited in a relatively infrequent manner under certain situations, such as a setting of terrorism response tasks, or emergent cardiopulmonary resuscitation (CPR) tasks.

For example, non-medical trainees on a space flight need to rapidly perform an advanced cardiac life support task during space flight missions (Ramos & Chen, 1994). This infrequent and unexpected situation requires precise and expeditious responses with assured quality of performance. To guarantee high human performance with non-failures and maximized readiness while minimizing the resources required to maintain this state, the phenomenon of learning and forgetting mechanisms should be taken into consideration.

Similarly, workers in advanced manufacturing enterprises, who need to acquire a wide variety of knowledge and skills, can exhibit critical performance degradation under overloaded and cognitively demanding tasks. Because a wide array of acquired knowledge and skills would be elicited in an infrequent manner, the disuse or infrequent use of knowledge can lead to human performance decrements resulting from knowledge degradation.

The global market challenges workers to be multifunctional by learning the greater variety of skills. The multifunctional workforce requires acquisition of various skills. To address the challenges, it is necessary to understand attributes skills for training and training regimens that can manipulate workers' knowledge structures to increase knowledge and skills retention.

A report sponsored by National Science Foundation (Division of Design, Manufacture, and Industrial Innovation) ranked research needs associated with integrating humans in advanced manufacturing systems for the U.S. manufacturing industries, to maintain competitiveness under the global economy (Mital, 1995). Especially, human issues of training (e.g., cross-training or retraining) were identified as a key research need for preparing the U.S. workforce to be viable and competitive in the global competition.

Nevins and Whitney (1989) acknowledge that trends in automation have led to automation of simple and monotonously repetitive tasks, while leaving complex unique tasks for humans. In particular, the introduction of automation has placed greater cognitive loads (e.g., analysis, decision-making, problem-solving, and judgment) on workers while reducing physical loads. Thus, Mital (1997) insists that humans still remain as a considerable and increasingly important element in advanced manufacturing systems.

Radical changes in manufacturing environments necessitate that workers should acquire a wide array of knowledge and skills at various levels to produce marketable products (Mital et al., 1999). However, task complexity has significant effects on the variance of individual learning and forgetting rates (Nembhard & Osothsilp, 2002).

Nembhard and Osothsilp (Nembhard & Osothsilp, 2002) investigated learning and forgetting effects on a manufacturing task that is related to sewing tasks with a worker-paced machinery at a textile manufacturing plant. The task requires a high level of hand-eye coordination and manual dexterity. Their findings indicate that workers, at higher task complexities, are more variable in their learning rates, forgetting rates, and productivity rates than they are at lower task complexities.

In technological manufacturing systems, high error rates (e.g., human performance discrepancy, accidents, malfunction of machines, inferior quality, lessened productivity, etc.) can be dramatically reduced to a minimum level when workers effectively utilize accumulated experience (Duffey & Saull, 2003). This highlights the importance of training to assure that knowledge and skills are resistant to decay.

Disuse or infrequent use of knowledge and skills can aggravate degradation of knowledge and skills, and is likely to produce poorer human performance. The acquired knowledge and trained skills for responding to unexpected terrorist attacks would be susceptible to this forgetting phenomenon. “What has been learned gets forgotten” is a general human characteristic.

“Engineered” learning can reduce loss of acquired knowledge and skills, thereby decreasing unwanted and unexpected errors. One can only speculate the overall loss in

U.S. productivity due to the degradation and erosion of critical, infrequently used skills by the workforce. As such, "readiness and preparedness" for overall productivity through a novel paradigm of training is essential to enable workers to efficiently respond to unexpected situations in advanced manufacturing systems.

Understanding and, ultimately, ameliorating productivity loss due to skill decay requires a valid fundamental model of the mechanisms responsible for this decay, and methods for counteracting these mechanisms. Training programs for industry workforce and military warfighters in the United States based on the "engineered knowledge" paradigm will advance strategies for preparedness and readiness.

1.2 Objectives

The objectives of the thesis are to provide a novel training paradigm to better shape the workforce, and to quantitatively measure training performance and proficiency. There are three goals to achieve the objectives.

First, acquisition and degradation phenomena will be modeled with the ACT-R architecture. The cognitive model will be expected to provide a quantitative measure of training performance. Second, attributes of knowledge and skills with respect to degradation will be investigated. Finally, based on the identified knowledge attributes, possible mitigating factors that can inoculate knowledge and skills against decay will be explored and utilized to instantiate decay-resistant knowledge principles. Figure 1.1 shows the research goal and approaches.

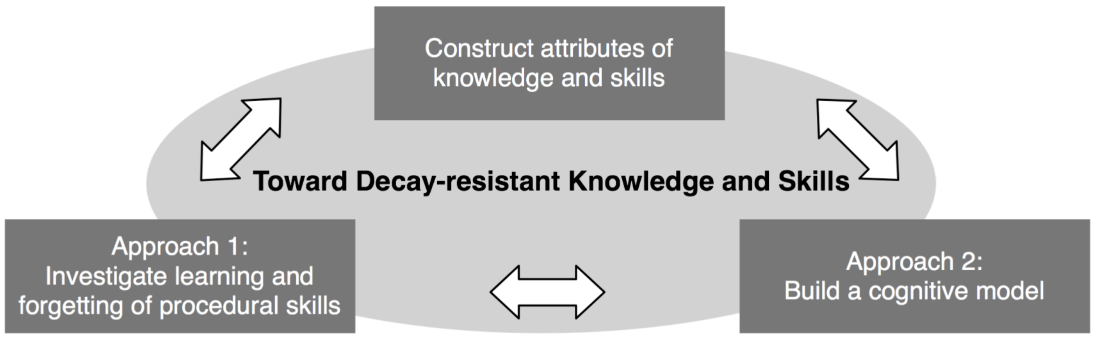

Construct attributes of knowledge and skills
Toward Decay-resistant Knowledge and Skills
Approach 1: Investigate learning and forgetting of procedural skills
Approach 2: Build a cognitive model

Figure 1.1. A schematic representation of the research goal and approaches.

1.3 Research Contributions

The scope of the thesis project encompasses both theoretical and practical contributions including measuring training efficiency, optimizing training programs, and understanding knowledge and skills attributes that can be resistant to decay. This thesis produces contributions to support a paradigm shift of designing a training program in industry and military and human performance modeling of a complex task using the ACT-R architecture.

It is of interest to delve into the leading-edge technology of modeling human performance. New quantitative methods such as Soar and ACT-R, known as cognitive architectures, are of interest. These tools are viewed as an emerging technology from cognitive science that can be applied to broader engineering issues (Byrne & Gray, 2003; Pew & Mavor, 1998).

For example, Mertz (1997) studied how the simulated agent built in the Soar cognitive architecture can be used to design instructional lessons for training circuit board assemblers. He stated that operator control knowledge in Soar is well defined by using a context, an operator, and a preference. The context indicates the working memory state where knowledge expressed in production rules is to fire. The operator represents a primary unit of cognition. The preference represents the desirability to apply an operator and provides a selection rule to choose what operator will be fired to create the next state. Thus, Soar's learning mechanism provides a backward benefit to determine what the content of instruction would be if we assume that operator control knowledge gives us what assemblers need to learn. Based on the emerging techniques of modeling and simulating human performance, ultimately, scientific management of training programs could be provided to shape and steer vital workforce members in military and industry contexts.

1.3.1 Contributions to Training

In manufacturing enterprises, it is necessary to shorten the time to market for a new product with the best quality. Based on the accumulated knowledge base, a company would utilize know-how to manufacture a product. Engineering the product knowledge base is critical factor to successful competition in the market.

Similarly, a human will acquire and accumulate knowledge in memory. To effectively retrieve and use knowledge, it is critical to engineer the acquired knowledge. The “engineered” knowledge examined here is anticipated to enhance human performance in high-stakes industry or military situations involving infrequent use of acquired knowledge and intermittent training. In addition, a learning and forgetting model using a cognitive architecture will provide a theoretic and scientific understanding of knowledge and skill processes.

The greater the forgetting of knowledge and skills, the more drills would be essential to retrain a large number of first responders repetitively. Due to the real-world constraints of resource availability, it is impossible to train every personnel on every task to the degree that is necessary to minimize knowledge degradation (Hagman & Rose, 1983). Even though sufficient training would be necessary, it may not be economically acceptable to perform $16 million training drills weekly or biweekly across the whole nation. This would result in enormous waste of fiscal assets. Insufficient and inappropriate training that can induce knowledge degradation will be a crucial factor to increase performance discrepancy and latency of response. Therefore, implementation of strategic training programs is necessary to inoculate knowledge against decay.

As a broader impact, the emergency responders will get tangible and practical benefits from research efforts to develop a theoretical model of learning/decay mechanisms of knowledge. Particularly, a cognitive model will be dedicated to establishing training principles to inoculate knowledge against decay. Instantiation of decay-resistant knowledge principles has a captivating intellectual merit. The principles can be applied to implement a novel training paradigm and train the workers to be viable (e.g., strategic retraining training to reduce knowledge and skills degradation). Furthermore, a new paradigm of engineered knowledge training will advance strategic and cost-effective training plans for the U.S. industry and military.

1.3.2 Contributions to Human Performance Modeling

First, the ACT-R model is implemented to emulate a set of complex spreadsheet tasks. In particular, a model of skills retention that is of primal interest in this thesis study produces applied implications to model human performance. Second, steps are taken to provide the skills retention model is embodied to directly interact with the task environment, providing a more realistic analysis of the model behavior. This will enable more advanced comparison with human performance.

1.4 The Dissertation Outline

Figure 1.2 shows the overview of the dissertation. Chapter 1 contains a brief introduction to the research topic, problem statements, and brief summary of research contributions. Chapter 2 contains theoretical synopsis of relevant findings about skill acquisition and degradation. Based on the findings, I will describe how I pursue and investigate modeling and simulation of skills degradation. Chapter 3 contains detailed information on the cognitive architecture, ACT-R 6, and the model developed to explain the theory developed in Chapter 2. The skill retention model and its performance are explained in this chapter. Chapter 4 contains a study with human subjects to explore the details of knowledge and skills degradation. Overall task performance and subtask performance are both analyzed. Chapter 5 contains comparisons of the model and human performance. Chapter 6 contains a discussion and the conclusions of the thesis.

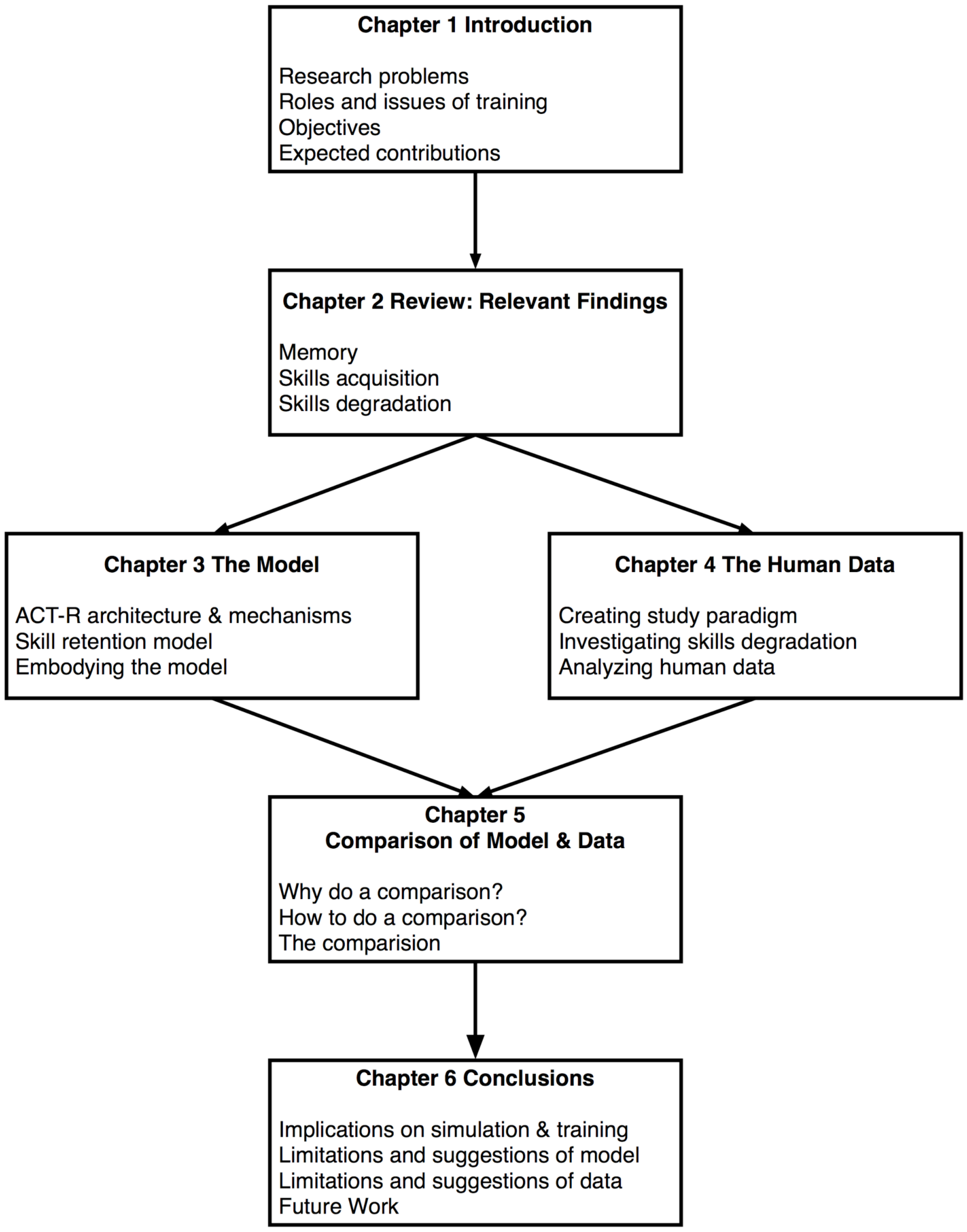


Figure 1.2. Overview of the dissertation study.

Chapter 2
Review: Knowledge and Skills Acquisition, and Degradation

In this chapter, I am exploring fundamental and theoretical foundations of how people learn and retain knowledge and skills in memory. In particular, relevant findings in cognitive science and engineering are reviewed to pursue scientifically reliable investigations on skills degradation. Based on the extensive review, I will describe my thesis research design to explore and to test skills degradation.

2.1 Knowledge and Skills in Human Memory

I learned the Miller's magical number seven plus or minus two in an undergraduate level course about human factors. At that time, the instructor emphasized a human's working memory capacity, giving an example of the seven digits phone numbers everyday we use such as 237-7381. The seven digits help people to store the phone number in human's memory.

The term of working memory appears to have been first proposed by Miller and his colleagues (Miller, 1956; Miller, Galanter, & Pribram, 1960), and has been used to describe a system representing temporary maintenance and manipulation of information (Baddeley, 2001). Also, we can encounter the term of working memory in computational modeling domains (Lovett, Reder, & Lebiere, 1999; Newell & Simon, 1972; Young & Lewis, 1999).

Atkinson and Shiffrin (1968) proposed that the human memory system can be represented by a sensory register, a short-term store, and a long-term store. The sensory register receives external inputs from the outside world. Registered external inputs are transferred to the short-term store that is viewed as a working memory system. It is assumed that the information entered into the short-term store is subject to decay. Then, the information can flow to the response generator or be transferred to the long-term memory. In this context, the working memory is thought of as a simple storage of information.

Baddeley and his colleagues have elaborated the simple storage of working memory and proposed a concept of the multi-component system that has been used over three decades (Baddeley, 2001; Baddeley & Hitch, 1974). In this working memory system, the "central executive", an attentional controller with a limited capacity, is interacted with two other components of the phonological (articulatory) loop and the visuospatial sketchpad.

The first component of the phonological (articulatory) loop is concerned with acoustic and verbal information. The phonological loop system is assumed to retain verbal information over a short period of time, consisting of a phonological store that holds phonological information, and an articulatory rehearsal system that serves to maintain decaying representations in that phonological store (Baddeley, Gathercole, & Papagno, 1998). The second component of the visuospatial sketchpad is assumed to maintain visual and spatial information.

Finally, the component of the central executive was initially the vaguest system, serving as a ragbag filled with awkward questions such as what determines when the phonological loop and visuospatial sketchpad are used and how they interact with each other in complex strategy selection, planning, or retrieval (Baddeley, 1996, 2001).

After the mid 1980s, Baddeley and colleagues begun their attempt to describe the central executive in detail, relying on Norman and Shallice's Supervisory Attentional System (SAS) that explains attentional control of action (Norman & Shallice, 1986). The potential subprocesses of the central executive are focusing attention, dividing attention, and switching attention. It is important to note that the memory is subject to alteration, and that the originally acquired memory can be lost (Loftus & Loftus, 1980).

2.2 Knowledge and Skills in a Realistic Task

Koubek, Benysh, and Tang (1997) explored the various types of knowledge and skills in the workplace and the changes of their acquisition with expertise. They discussed three major types of knowledge including declarative, conceptual, and procedural knowledge. These are noted in Table 2.1.

Declarative knowledge indicates factual information. For instance, when a user is doing a summation task in a spreadsheet, he/she needs to know the command. In the

Dismal spreadsheet, a user needs to use the “dis-sum” command to sum numbers. The knowledge of “dis-sum” is considered to be declarative knowledge.

Table 2.1. Types of tasks with respect to different knowledge and skills.

Task Types	Task Characteristics	Types of Knowledge	Examples
Procedural	Knowledge retrieval (information recall), Several decision-making points	“Procedural knowledge” and “declarative knowledge” (Anderson et al., 1998) “how-to-do-it knowledge” (Kieras, 1997)	Air traffic control task
Cognitive	Combining and evaluating incoming or acquired information, Making decisions	Cognitive-procedural knowledge (generally known as problem solving skills)	Programming Solving problems
Perceptual-motor	Perception and motor execution	Motor-procedural knowledge	Riding a bicycle Moving a mouse

Conceptual knowledge (relational knowledge) represents the core concept of a specific domain and the interrelations between the concepts (Koubek, Benysh, & Tang, 1997). For example, the statement of “the summation command in Dismal spreadsheet is dis-sum” indicates the relation of “summation command” and “dis-sum”. The conceptual knowledge consists of two or more items of declarative knowledge.

Procedural knowledge indicates knowledge representing human behavior. Another term for this knowledge classification is described as “how-to-do-it” knowledge (Kieras, 1997). This can be typically modeled using production rules. Production rules consist of the pairs of “condition” and “action” represented by IF/THEN rules.

Now, let us consider knowledge and skills in a task. Table 2.1 describes several types of task consisting of different knowledge and skills. Sabol and Wisher (2001) categorized a task into three components: *knowledge, decision,* and *execution*. Characteristics of a task can vary with regard to which components dominate in the task. For example, to accomplish a military task, a soldier should be able to: (a) retrieve knowledge and skills from memory; (b) evaluate a situation, combine incoming information, and decide alternative courses of action, and (c) execute the chosen procedural steps in a sufficiently skilled manner. In terms of the three components, a task can include procedural, cognitive, or perceptual-motor subtasks.

Surveys showed that personnel in technical jobs perform mostly procedural tasks (Tarr, 1986). For example, in an emergency situation, the most important knowledge and skills would be procedural, such as cardiopulmonary resuscitation (CPR) or a decontamination task of biological/chemical agents. Konoske and Ellis (1991) stated that procedural tasks consist of an ordered sequence of steps or operations performed on a single object or in a specific situation to accomplish a goal. During the accomplishment of a goal, a procedural task would involve several decision points. With regard to perceptual-motor task, "riding a bicycle" can be a simple example.

Hagman and Rose (1983) mentioned that the best predictor of forgetting is the number of steps required in the procedural tasks. There is a supporting study of skill retention conducted by the US Army Research Institute (ARI) during the mobilization of the individual ready reserve (Sabol & Wisher, 2001; Wisher, Sabol, Sukenik, & Kern, 1991). This investigation showed that procedural (discrete) skills might be forgotten much more rapidly than perceptual-motor (continuous) skills. However, it seems that we rarely forget how to ride a bicycle or how to swim after learning these skills. These are perceptual-motor control skills. This aphorism and their investigations suggest that procedural (discrete) skills might be forgotten much more rapidly than perceptual-motor (continuous) skills or that these skills are so overlearned.

It is presumed that knowledge types will affect acquisition and retention of knowledge and skills. With respect to the types of skills in a task, different acquisition and retention performance would be expected. Thus, investigating knowledge types will provide a scientific account for the necessity of different training regimens with respect to knowledge types. Realistic tasks in this section have a large procedural component and an interface. In this study, focus needs to be given to procedural tasks comprised of cognitive and perceptual-motor skills.

2.3 Knowledge and Skills Acquisition

Proctor and Dutta (1995) provided a good review of skills acquisition such as theories proposed by Fitts (1954, 1964), Anderson (1982), and Rasmussen's research (1986). Fitts distinguished three phases of learning: cognitive, associative, and autonomous. Based on this classification, John Anderson developed a theory of cognitive skill acquisition. As links of Fitts' phases, Rasmussen also proposed a framework pertaining to skilled performance that is differentiated by knowledge-based, rule-based, and skill-based. In this section, I explore these learning theories and their implications.

2.3.1 How Are Skills Acquired?

As humans learn how to perform a task, the time for completing that task would get faster and faster with practice. Learning behavior of humans follows a regularity known as a Power law of learning (Ritter & Schooler, 2001). The learning curve provides fundamentals both in cognitive psychology and in cognitive science. In cognitive psychology, the curve provides a mathematical account of the learning rate. Also in cognitive science, mechanisms representing the curve are used to build cognitive models that produce human behavior (Ritter & Schooler, 2001).

ACT-R's history can be found in a recent book, *How can the human mind occur in the physical universe?* (Anderson, 2007). The origins of ACT-R can be started from the book, *Human Associative Memory* (Anderson & Bower, 1973). Anderson and Bower proposed a symbolic representation of declarative memory. Anderson combined HAM's declarative system and Newell's symbolic procedural system into the first version of the ACT theory (Anderson, 1976). Also, he proposed subsymbolic systems: an activation quantity for declarative memory and a strength quantity for the procedural system. In 1983, he published a book pertaining to the ACT* system (Anderson, 1983). In this book, he proposed a goal-directed processing that is the root of the current ACT-R's goal module, and a concept of production learning (e.g., composition and proceduralization), which is the root of the current ACT-R's production compilation mechanism. With the influence of rational analysis, ACT* evolved into ACT-R (R denotes rational) and the first implementation of the comprehensive theory, built in Lisp, was available to the public (Anderson, 1993). Meyer and Kieras's (1997) EPIC system gave an influence of the perceptual-motor system to the ACT-R system (Byrne & Anderson, 1998). These days, ACT-R is evolving by the study of brain imaging with fMRI, such as mapping brain regions to ACT-R's modules (Anderson, 2007; Anderson et al., 2004; Qin et al., 2004).

For the framework of skill acquisition, Anderson (1982) proposed a clear distinction of two stages: a declarative stage and a procedural stage based on his earlier work of the ACT production system (Anderson, 1976). Koubek et al. (1999) reinforced that learning mechanisms in terms of human memory and cognition can be described by the theory of the Adaptive Control of Thought (ACT) proposed by Anderson in 1976.

ACT-R assumes that humans have various knowledge structures with a set of parameters (Anderson & Lebiere, 1998). There are two dimensions of learning mechanisms of ACT-R. One dimension is concerned with the acquisition of declarative and procedural knowledge. The other dimension addresses symbolic and subsymbolic learning. Symbolic learning is associated with the acquisition of the chunks and productions, whereas subsymbolic learning includes the acquisition of the parameters directing the knowledge elements. The ACT-R's learning mechanisms can also be used to understand the aspects of certain task types usually found in the problem-solving domain of advanced manufacturing settings (Koubek, Salvendy, Tang, & Brannon, 1999).

In the declarative stage, humans learn knowledge and skills from instructions. Acquiring information is considered as initial encoding of facts about the skill. Then, acquired information is interpreted to produce behavior. Through a mechanism called knowledge compilation, the acquired knowledge is converted to a procedural form with appropriate practice. After the knowledge compilation, further tuning of the knowledge occurs, producing speedup of the knowledge application process. This is referred to as a procedural stage of knowledge.

The relationships between the declarative and procedural stages are explained through the framework of the ACT production system where declarative knowledge is represented as a propositional network of facts and procedural knowledge is represented as productions. Each production has a condition/action rule and specifies when a cognitive act should take place (Anderson, 1982). Anderson stated that a set of productions is sequentially applied in a task in an attempt to represent the cognitive steps taken in performing the task. Thus, each production is viewed as a step of cognition in our mind.

Anderson (1982) asserted that the production system has important features to represent human learning theory. First, a condition of a production rule must be matched in terms of information in working memory that is a part of the ACT system's declarative component. Second, productions are hierarchically organized by a goal structure with subroutines associated with the goal to be achieved. This goal structure is fundamental to human cognition.

The mechanisms of knowledge compilation are broadly represented in ACT-R by composition and proceduralization. The proceduralization mechanism is further divided into the generalization, discrimination, and strengthening mechanisms. The composition mechanism is that sequences of productions that produce behavior can be collapsed into a single production that produces the same behavior, producing a speedup of performance. The composition process reduces the number of productions to be administered in performing a task. While the composition process creates productions that are domain-general, the proceduralization mechanism, then, constructs productions that no longer need domain-specific information to be retrieved from working memory. The composition and proceduralization processes together convert declaratively encoded knowledge into production form.

The composition mechanism itself is not sufficient to explain further speedup of performance by practice after learning of skills. The further tuning of skill acquisition can be explained by the aforementioned proceduralization mechanism comprised of a set of generalization, discrimination, and strengthening processes. The generalization broadens the applicability of production rules. In the meanwhile, a discrimination process lets productions become narrower. The strengthening process serves to have better rules strengthened and poorer rules weakened.

Based on the mechanism of knowledge compilation, the production compilation mechanism was later proposed to model complex skill acquisition within the ACT-R architecture (Taatgen & Anderson, 2002; Taatgen & Lee, 2003). Production compilation combines both proceduralization and composition mechanisms into a single mechanism.

By eliminating a condition, two rule productions are combined into a single rule. That is, a compiled rule is generated by eliminating the retrieval request in the first rule and the retrieval condition in the second rule (see Taatgen & Lee, 2003). Taatgen and Lee (2003) explained that the process is slow to retrieve a chunk from declarative memory because only one memory can be retrieved at a time. Production compilation allows a speed-up process by generating task-specific procedural knowledge.

2.3.2 Structuring Knowledge and Skills

As mentioned before, practice leads to skilled and fluent performance, producing a speedup of task completion time. The learning theory of the production compilation mechanism addresses the question of how practice increases the speed of performance. That is, restructuring of knowledge and skills is attributable to improving task performance.

Carlson and Lundy (1992) empirically studied learning of cognitive procedural sequences in terms of consistency in sequence operations and input data for those operations. One of their findings is that consistent practice with consistent sequence operations and data produced greater speedup in solving mathematical equations than varied practice. Also, they mentioned that learning by restructuring, that is, sequential calculations were replaced by memory retrieval, appeared to occur only with consistent data. Restructuring did not depend on consistency of sequence operations, but practice with consistent sequence allowed participants to take advantage of consistent data.

2.3.3 Skills Acquisition in Industrial Tasks

Nembhard and Prichanont (2007) stated the importance of creating multifunctional workforce in industry because the shortened life cycle of products requires industrial production systems to cope with the greater variety of products in less time. To meet these market challenges, workers need to acquire variety of skills from cross training. Furthermore, understanding of forgetting skills can help us to provide possible solutions to cope with the challenge.

Koubek et al. (1999) investigated a theoretical model of human skill acquisition in the domain of advanced manufacturing technologies. This study compared a variety of existing theoretical approaches such as learning theories, dual-processing code theory (automatization), cognitive resources theories, and knowledge structures. Based on these theoretical approaches, a hybrid model was proposed. This hybrid model describes the interactions of existing knowledge acquisition models. In the proposed model, learning hierarchy, automatization, and cognitive resources are all interrelated in skill acquisition processes. More cognitive resources are required for moving up the learning hierarchy to learn a new skill. On the other hand, repetition of consistent tasks gradually releases cognitive resources through automatization. Because learning requires cognitive resources, a lack of these resources inhibits movement up the hierarchy of learning.

Koubek and Salvendy (1991) proposed three levels of knowledge structures, which account for the determinants of performance in skilled cognitive tasks. The knowledge structures hierarchically include behavioral outcome of knowledge, processing of knowledge, and structure of knowledge. These knowledge structures can be manipulated through different types of training as a factor influencing skill acquisition (Koubek, Clarkston, & Calvez, 1994). The results of this study suggest that there is a significant effect caused from the sequence of training material, contrasting a top-down versus bottom-up approach. Fundamental motivations were generated from the study to further delve into the subsequent mapping of task characteristics to knowledge structure types, task requirements, and knowledge structure dimensions.

2.3.4 Spacing Effects of Knowledge Acquisition

A learning process can take place in a massed or distributed way. A massed practice approach (MPA) indicates a training set occurring closely spaced in time. A distributed practice approach (DPA) indicates a set of learning processes that is distributed in time. Variety of intervention intervals affects performance of massed or distributed practice. The intervention interval can be represented by a frequency of learning and practice activities. If the frequency is high at the beginning of a learning session, it is thought of as a massed practice. If the frequency of learning sessions is low and uniformly distributed, the learning and practice can be considered as a distributed practice.

Typically, it is said that a distributed practice approach would require less learning time and provide longer knowledge retention periods. Research has shown that a group with MPA took 51% longer than another group with distributed practice to complete the task and committed 40% more errors, even though the overall error rate was low for both groups (Hagman & Rose, 1983).

As an extension of the distributed fashion of learning, the spacing effect of learning is a general phenomenon of the memory research domain. Table 2.2 shows a brief review of literature with respect to several spacing effects. The majority of published studies of spacing effects have examined memory for declarative knowledge rather than procedural knowledge.

In particular, Pavlik and Anderson (2003, 2004, 2005) studied the spacing effects and optimization of learning. The spacing effect is considered as the memory benefit that controlling the time duration between practices can enhance human performance. Pavlik and Anderson investigated the evidence of spacing effects to learn Japanese-English pairs in declarative memory. From the paired-associates experiment, an activation-based model was proposed. This model proposes that an item receives an increment of strength when it is practiced, but that these increments decay as a power function of time.

To sum up, knowledge and skills are acquired from instructions in the declarative stage. This encoding of information produces knowledge structure and is interpreted to produce behavior. Practice of acquired knowledge and skills can produce speedup of performance by restructuring. In addition, spacing of learning can affect performance. Important factors for learning include instruction, knowledge structure, practice, and spacing of learning. Thus, these factors are given consideration to understand and control acquisition of knowledge and skills.

Table 2.2. Summary of spacing effects literature.

Domain	Knowledge Type	Findings	Reference
English-Spanish word pairs learning	Word	The study investigated the effects of the time interval separating reacquisition sessions (1, 7, and 30 days). One finding is that the cumulative effect of the intersession interval is relatively small. If knowledge retrieval interval is much longer than the intervals of reacquisition sessions, much of the acquired knowledge is not well accessible.	Bahrick (1979)
Paired-associate learning	Word	Less learning occurs when repetitions of an item are massed than when they are distributed	Greeno (1964)
Verbal recall test	Monosyllabic nouns	A certain degree of overlearning is economical to retention performance for intervals of 2 to 28 days.	Kruger (1929)
Picture recall test	Pictures of objects (e.g., airplane, arrow)	As long as the number of prior tests is held constant, there is no evidence that recall increases as the retention interval is increased. If the retention interval is held constant in the meanwhile the number of prior tests increases, recall increases.	Roediger & Payne (1982)
Paired-associate learning	Paired-associate and recognition memory	Response recall is functionally related to the spacing of repetitions as a function of the retention interval. At a short retention interval, the spacing function is nonmonotonic. The spacing function increases monotonically at longer retention intervals—a short retention interval: 2 and 8 intervening events, a longer retention interval: 32 and 64 intervening events, each event was visible for 3 sec.	Glenberg (1976)
Paired-associate learning	English-foreign language word pairs	Extended retrieval practice of foreign language vocabulary learning produces significant retention benefits over a 5-year retention period after the termination of training. These benefits are maximized under the 2-month interval between retrieval sessions—relearning sessions were conducted at intervals of 2, 4, or 8 weeks. Retention was tested for 1, 2, 3, or 5 years after training terminated.	Bahrick et al. (1993)

2.4 Knowledge and Skills Degradation

This section provides a brief synopsis of knowledge and skills degradation both in psychology and operations research. From the perspective of psychology, forgetting is explained by theories and mechanisms that can address varying types of knowledge and skills. From the perspective of operations research, one attempts to predict productivity based on workforce performance of learning and forgetting. Both perspectives can help us to understand the issues of interest in this dissertation study.

2.4.1 Synopsis of Skills Degradation in Psychology

Knowledge and skills acquired through certain types of education or training can be forgotten with the passage of time. Forgetting, or knowledge decay, can induce discrepancies in trainee performance. Sabol and Wisher (2001) presented the reasons for the discrepancies by stating three types of decreased abilities: (a) inability to retrieve knowledge from memory, (b) inability to perform cognitive processing (e.g., making a decision, or selecting tactics, etc.), and (c) inability to execute an action or procedure in a skilled manner.

The first ability is important to perform procedural tasks because these tasks rely largely on acquired knowledge retrieval (Sabol & Wisher, 2001). However, procedural memory tends to undergo decay over time. The memory of cognitive processing also suffers from a moderate rate of knowledge decay (Cooke, Durso, & Schvaneveldt, 1994). Sabol and Wisher (2001) classified execution of skills into continuous and discrete skills. For example, continuous skill can involve riding a bicycle, and the discrete skill can be executed to disassemble a carburetor. Evidently, discrete skills are more susceptible to degradation. This coincides with a study by McKenna and Glendon (1985) that found that less than a quarter of all trained personnel were skillful at performing the first aid task of cardiopulmonary resuscitation (CPR), six months after training.

Anderson and Neely (1996) define interference as the impaired ability to remember an event when it is similar to other events that are stored in memory. Thus, interference can occur when stored episodes or knowledge are blocked and are not recalled due to the intrusion of similar episodes or knowledge. The basic attribute of interference theory is retrieval cues. A retrieval cue is an associative link among stored items in memory. Interference can be attributable to skill decay, indicating a retrieval cue that is available at the time of recall fails to access the target memory.

Interference can be classified into proactive and retroactive interference, given that there are two tasks (Task I, Task II) to be learned through training. Proactive interference indicates that the previously learned knowledge (Task I) inhibits the retrieval of the learned knowledge of Task II. Conversely, the retroactive interference indicates that the source of interference is Task II that is learned later.

There is a difference between proactive interference and negative transfer. Negative transfer occurs when the acquired knowledge of Task I retards the successive learning process of Task II. In the mean time, the proactive interference occurs after both Task I and Task II are to be learned (Anderson, 1995). It is reasonable to conclude that negative transfer can be directly related to the rate of learning.

Efforts have been put forth to computationally model and simulate procedural knowledge degradation. The aforementioned variables including interference and negative transfer are fundamental factors to understand decay mechanisms. Brannon and Koubek (2001) speculate that variables including interference can diminish the ability to retrieve procedural knowledge for tasks in the domain of product assembly or supervisory control.

Cognitive psychologists and scientists have been trying to propose mechanisms of forgetting (e.g., interference) and to model some forgetting behaviors (e.g., forgetting paired-associate vocabulary, see Section 2.6.3). These efforts helped to acknowledge issues of skills decay by different types that can be continuous or discrete, or perceptual-motor or cognitive, etc. Besides the domain of cognitive psychology and science, there also has been on-going research on forgetting from the perspective of operations research. This will be discussed in the next section.

2.4.2 Synopsis of Skills Degradation in Operations Research

Engineers in operations research have studied workers' learning and forgetting to predict productivity and performance (see Dar-El, 2000; Nembhard & Prichanont, 2007). The industrial learning and forgetting curves are used to estimate labor cost or cost of strikes. Break length is one of the factors causing a lack of skill retention. Once there is a sufficient break length, forgetting processes are triggered. Worker sickness, vacation, or strikes leads to dormancy of activity, and the latency eventually results in the phenomena of forgetting. Consequently, the forgetting curtails productivity and leads to inferior quality of products. Therefore, forgetting necessitates continuous learning of knowledge and skills for workers to maintain productivity.

In psychology, the Power function is generally accepted to well describe simple human learning, assuming replacing incorrect response tendencies with correct ones. Unlikely, Mazur and Hastie (1978) argued that a hyperbolic function well describes the learning process because it is a process of accumulation where incorrect response tendencies remain constant and correct response tendencies increase with practice. The hyperbolic function is shown in Equation 2.1. The amount of learning is represented by y, the amount of time (or training) by t, and the asymptote for learning by k. R determines the rate of approach to the asymptote of k.

$$y = k\left(\frac{t}{t+R}\right)$$ **Equation 2.1**

Nembhard and Uzumeri (2000a) investigated the fitness of several published learning curves such as log-linear, hyperbolic, and exponential curves, by using multi-criteria comparison. The comparison criteria include the variance of the fit, the number of parameters, and the ability to capture episodes of negative learning behavior. The best model is a hyperbolic function with three parameters shown in Equation 2.2. y is a measure of work performance, x is the amount of cumulative work in units of time (or trials), and p is the individual's accumulated prior experience for that task with the same unit as x. The parameter, k, indicates an estimate of asymptotic limit. That is, the parameter represents the expected maximum performance level after learning. The parameter, r, indicates the cumulative production to achieve an output of $k/2$, indicating small values of r means learning occurs rapidly.

$$y = k\left(\frac{x+p}{x+p+r}\right),$$ **Equation 2.2**

subject to, y, k, p, $x \geq 0$ and $p + r > 0$

This hyperbolic function model consisting of three parameters produces the best performance (Nembhard & Uzumeri, 2000a). That is, the model has the lowest average of the mean-squared error (*MSE*) statistics, $\hat{\mu}_{MSE}$ = 0.00577, indicating flexibility to capture the various learning shapes from the different individuals. The variability of the goodness-of-fit, $\hat{\sigma}_{MSE}$, from the hyperbolic model has the lowest value, indicating the model fits across individuals more consistently. Also, the number of parameters, which is three for the hyperbolic model, provides parsimonious representation of the learning curve (Nembhard & Uzumeri, 2000a).

In general, using a forgetting model with a decay function (e.g., an exponential function) requires a known value of the time of the break. The hyperbolic function can represent negative learning when r is less than 0. Based on this property, Nembhard and Uzumeri (2000b) proposed a model of learning and forgetting. They state that a practical forgetting model should be able to capture multiple breaks at irregular intervals. They modified the hyperbolic function of the learning curve by incorporating experiential learning (R), called *recency*, as shown in Equation 2.3. x in the denominator indicates each unit of cumulative production, and $t_x - t_0$ indicates the elapsed time for the unit x that is the difference between the time stamps of the completion of the current unit and the start of the first unit (t_0). Thus, Equation 2.3 represents the *recency* measure, the ratio of the average elapsed time to the elapsed time of the most recent unit produced.

$$R_x = \frac{\sum_{i=1}^{x}(t - t_0)}{x(t_x - t_0)}$$ **Equation 2.3.**

In this model, R_x was bounded below by 0 and above by 1. A value approaching 1 indicates all experience obtained immediately preceding the current unit, and a value approaching 0 indicates experience obtained infinitely long ago. For a constant production rate, the *recency* of R_x tends to a nominal value of 0.5. To decide the recency effect, the aforementioned cumulative production of x is discounted by the factor, R_x^{α}, shown in Equation 2.4, where α is an individual forgetting rates.

$$y_x = k(\frac{xR_x^{\alpha} + p}{xR_x^{\alpha} + p + r}) + \varepsilon_x \qquad \textbf{Equation 2.4}$$

Equation 2.4 includes both the learning rate of r and the forgetting rate of α. When we let α>0, small values of α produces little discounting of cumulative work. If the forgetting rate (α) increases, the term xR_x^{α} becomes smaller, indicating a greater discounting of the cumulative work (Nembhard & Uzumeri, 2000b). The investigation by Nembhard and Uzumari focused on examining the model performance to predict the amount of forgetting in intermittent learning/forgetting situations. According to this study, Nembhard and Uzumari (2000b) stated that the *recency* model provided efficiency of forgetting prediction by the low mean absolute deviation (MAD) and consistently stable performance by the low standard deviation of absolute deviations (STDAD).

Understanding human learning and forgetting has an intellectual merit. Those brief synopses about forgetting help us to comprehend the interesting questions: (a) how forgetting occurs? and (b) how forgetting can be predicted? Now, I am discussing what factors can affect and reduce forgetting.

2.5 Factors Supporting Retention

From human memory perspectives, there are several processes addressing forgetting phenomena such as decay, interference, and degradation. Decay is a forgetting process indexed by time and interference is a process indexed by the amount of distracting information (Altmann & Schunn, 2002). Historically, decay has been associated with forgetting in short term memory (Brown, 1958; Peterson & Peterson, 1959).

The term, “engineered knowledge”, is to represent “engineering” of knowledge degradation with time passage. We expect that training based on engineered knowledge and skills can augment human performance in spite of disuse of skills over time. Engineered knowledge and skills can be accomplished through identifying attributes of knowledge that are resistant to decay. That is, we might presume that decay rates of skills might be different by varying types of knowledge and skills. Also, spacing of training might have different forgetting rates.

Figure 2.1 shows theoretical constructs for the engineered knowledge training. The constructs consist of external and internal domains. For the external domain, industrial workers will be trained through knowledge acquisition tools such as original training, retraining, task-oriented training, or refresher training.

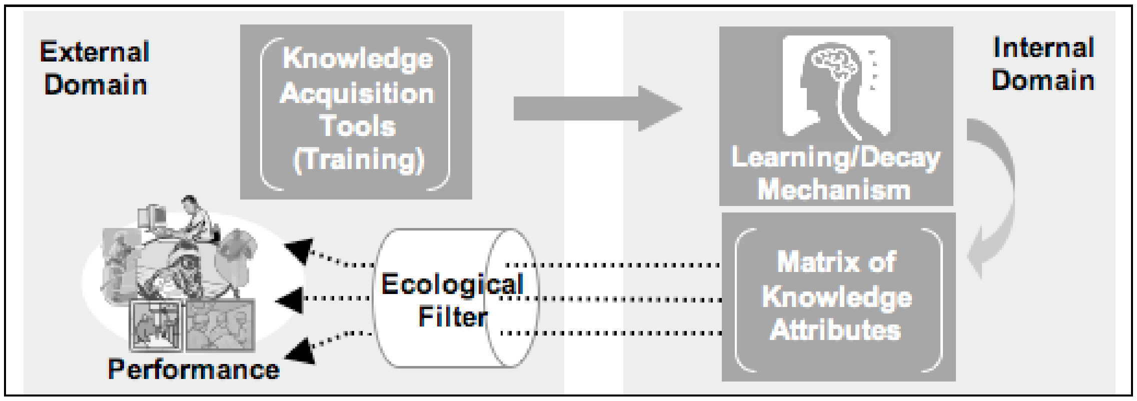


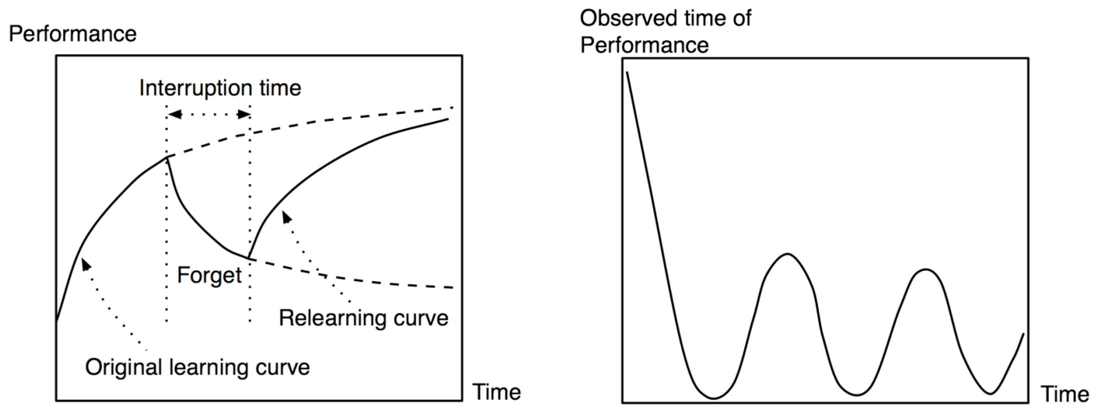
Figure 2.1. Constructs for the engineered knowledge and skills training.

Knowledge attributes (e.g., level of automatization, or robust knowledge structures) can be determined by both knowledge acquisition tools (e.g., training methods). Finally, performance in the external world will be exhibited by the retrieval and execution of internally stored knowledge through an ecological filter. The ecological filter incorporates task characteristics, psychological stress, rapid response (time-critical situations), and infrequent use of knowledge.

2.5.1 Training Factors for Retention

Original Training: Farr (1987) described factors influencing long-term retention of knowledge and skills. One important factor for long-term retention is the amount or degree of original (initial) learning. Overlearning (high-acquisition) can reduce the rate of decay because the amount of overlearning can increase the amount of knowledge acquisition.

Training and Retraining: Knowledge and skill retention that is resistant to decay can be described as the phenomenon of learning-forgetting-relearning (LFR) processes. Carlson and Rowe (1976) simplified the learning and forgetting curves by considering a log-linear pattern of time per unit. This approach allows the convenience of converting performance versus duration of experience (performance = standard time/actual time).



(a) Performance vs. elapsed time, Adapted from Carlson & Rowe (1976).

(b) Observed performance time with intermittent learning and forgetting periods, Adapted from Ramos & Chen (1994).

Figure 2.2. Learning-forgetting-relearning graphs.

The classical LFR model is shown in Figure 2.2 (a). From the LFR model, it is possible to know how much learning is lost if an asymptotic value of the maximum performance (representing fully learned or trained) is added (Dar-El, 2000). This asymptotic value can have important implications that help to understand the relationships between learning and forgetting. In the context of this proposal, the maximum performance level can be viewed as the desired readiness and preparedness of the workers for superior quality and productivity without performance decrements.

Unlike the Carlson and Rowe's graph, Ramos and Chen (1994), shown in Figure 2.2 (b), tried to integrate learning and forgetting in an attempt to provide a more realistic picture of what is actually occurring in those processes. The purpose of this integrated model is to establish training and retraining parameters such as amount of training and timing of retraining. In addition, they considered knowledge/skill acquisition and retention.

Figure 2.2 (b) shows the integrated learning and forgetting with intermittent periods between them. The expected performance varies in terms of time with a cyclic trend and does damp down towards a lower bound. In the Figure 2.2 (b)**,** the upward cycles indicate forgetting, and the downward cycles reflect learning.

Task-Oriented Training: Training can be classified into task-oriented and topic-oriented. Task-oriented training implies the use of context-based tasks to teach the factual knowledge and cognitive skills. On the other hand, topic-oriented training demands the information is taught more abstractly. The task-oriented trainees showed longer retention of knowledge and skills than did the topic-oriented trainees (Sabol & Wisher, 2001; Wisher, Sabol, & Ellis, 1999). High original learning for decay-resistant knowledge may be attained by conducting task-oriented training rather than topic-oriented training (Sabol & Wisher, 2001).

Training with Appropriate Retention Interval: Knowledge decay can be mitigated by appropriate retention intervals. Bahrick (1979) investigated how the maintenance of knowledge is related to the successive relearning sessions with varying time intervals. In this investigation, English-Spanish vocabulary pairs were tested with relearning sessions. Bahrick concluded that time between practice-sessions should be spaced at intervals not much shorter than the interval separating a practice from a test for optimum maintenance of knowledge. Retention performance can be increased by using appropriately “spaced” or “distributed” repetitions during practice sessions (Sabol & Wisher, 2001).

Training with Feedback: Farr (1987) mentioned that instructional strategies can play a significant role in mitigating knowledge decay. Training with feedback, which gives the trainee sufficient information to comprehend performance errors, is necessary to assure effective learning and enhanced knowledge retention.

Refresher Training: Retention of military knowledge and skills has been studied extensively (Sabol & Wisher, 2001; Wisher, Sabol, & Ellis, 1999). It was proposed that optimizing the schedule of “refresher training” can increase skill retention. The U.S. Army Research Institute for the Behavioral and Social Sciences (ARI) developed a

Trainer's Guide for Refresher Training with rankings of tasks in terms of the vulnerability to decay (see Wisher, Sabol, & Ellis, 1999, p.20). I report their findings here as an example of the refresher training guide for trainers.

Table 2.3. Ranking of task retention.

Rank	Task	% Go	Able to remember
#1	Extraction from minefield	0 %	↑ Hardest
#2	React to civilian on battlefield	8 %	
#3	React to sniper	9 %	
#4	Prevent shock	18 %	
#5	Carbon monoxide inhalation	28 %	
#6	Apply tourniquet	29 %	
#7.5	React to indirect fire	30 %	
#7.5	Winter driving	30 %	
#9	Vehicle search	34 %	
#10	Negotiation	36 %	
#11	Rules of engagement	42 % (27 %)	
#12	React to media	54 %	
#13	V corps convoy mine strike drill	56 %	
#14	Living in the cold	62 % (48 %)	
#15	Identify/detect trip wires	68 %	
#17	Driving postcheck	71 % (44 %)	
#17	Working in the cold	71 %	
#17	Identify/detect booby traps	71 %	
#19	Sleeping in the cold	73 %	
#20	Recognize/react to UXO	75 %	
#21.5	Mine detection	76 %	
#21.5	Locate a mine by probing	76 %	
#23	Driving precheck	89 % (62 %)	
#24	Personal search	90 % (62 %)	
#25	React to mines	96 % (68 %)	
#26	Field dressing/pressure dressing	98 %	
#27	Indications of mines/ booby traps	99 % (84 %)	↓ Easiest

Note: % Go indicates percent of soldiers predicted to perform the task at "Go" level after two months of skill disuse. Percentages in parentheses apply when job aids are unavailable.

2.5.2 Ecological Factors for Retention

Task Characteristics: Sabol and Wisher (2001) stated that knowledge decay from procedural memory is affected by the characteristics of tasks such as task complexity, task demands, and environment where a task to be performed. A complex task can contain a number of procedural steps to be done. Thus, as the number of task steps increases, the decrements of performance become more severe (Sabol & Wisher, 2001).

Psychological Stress: Berkun (1964) investigated human performance decrements under psychological stress. In this investigation, the term "psychological stress" was meant to imply the apparently existing threat to the survival of someone for whom she/he is responsible. One experiment represented an emergency crash landing of an aircraft where army trainees were passengers. The subjects were all young men engaged in their basic combat training. For retention of knowledge, the subjects under stress correctly recalled an average of 4.9 out of 12 answers while the subjects in the control group recalled an average of 7.6 answers. Stokes (1995) studied stress-resistant performance in aeronautical decision-making. Novices and experts performed cognitive tasks under stressed and unstressed conditions. Stress conditions included task loading, dual-task loading, time pressure, noise, and financial risk. The results showed that experts who have high preexisting anxiety or experience showed no performance decrements under stress. In contrast, the novice group was affected by the stress conditions while performing aeronautical decision making tasks.

2.5.3 Robust Skills with Enhanced Retention

Previous sections addressed learning and forgetting corresponding to internal factors for robust engineered knowledge. Knowledge acquisition tools and ecological constraints were presented in previous sections with respect to external factors. As discussed before, performance will be exhibited by filtering through ecological constraints via internal attributes of knowledge that is obtained by training tools.

Figure 2.3 summarizes the internal attributes of knowledge in the learning and forgetting. These attributes of knowledge can be manipulated by controlling knowledge acquisition methods and by understanding the fundamentals of learning and forgetting mechanisms.

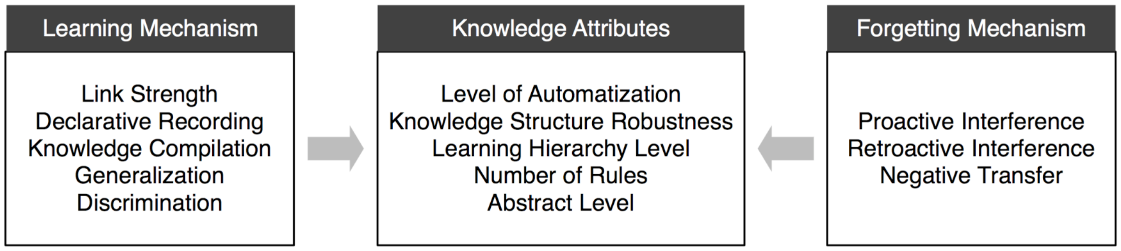


Figure 2.3. Knowledge attributes of learning and forgetting.

Investigating attributes of knowledge and skills by different retention intervals can produce useful findings to optimize long-term retention of knowledge and skills. Koubek, Salvendy, and Noland (1994) stated that knowledge structures can affect cognitive-oriented tasks and can be controlled to become more robust by manipulating knowledge acquisition methods. Therefore, engineered knowledge can be achieved by identifying knowledge attributes. The need for this optimization forced me to investigate knowledge and skill attributes in the context of learning and forgetting. If we know what nature of knowledge and skills causes decay, then we can engineer those factors to make skills more robust.

2.6 Models of Memory

This section reviews existing models of memory. Anderson and Schooler (1991) examined a number of environmental resources to determine whether human memory would be behaving optimally in terms of the pattern of past information presentation. Major patterns of past use of information can include how often information has been practiced (frequency) and how long it has been after the last practice (recency). In addition, successive repetitions of an item affect how well that item is retrieved from memory, which is called the spacing effect. Based on these environmental resources (e.g., the frequency, recency, and spacing effect), I am delving into models of memory to build a cognitive model that is able to learn and forget knowledge and skills.

2.6.1 Learning Models of Declarative Memory

Latency generally refers to the time delay between an input and an output in communications, operations, or simulations. An input can be an initiation of a request to a system. Then, the system is expected to provide an output with respect to that request.

Latency is a useful factor to measure learning effects of memory because it decreases as a power function of the number of practice trials. For example, Grant and Logan (1993) found that repetition priming increased as a power function of practice while it decreased as a power function of retention interval or delay. The power function has been representing the effects of practice or learning (see Anderson, 1982; Anderson, Fincham, & Douglass, 1999; Newell & Rosenbloom, 1981), but there have been discussions of other functions (see Anderson & Tweney, 1997; Myung, Kim, & Pitt, 2000).

Anderson, Fincham, and Douglas (1999) conducted a set of experiments to investigate practice and retention. Participants memorized eight different facts such as "Skydiving was practiced on Saturday at 5 p.m. and Monday at 4 p.m." The task includes an underlying rules about the time relationship between the two events for that sports. For example, the second skydiving event always occurred 2 days later and 1 hour earlier. After memorizing the facts, participants were tested with rule application problems. The task requires participants to retrieve the learned facts and to apply rules to achieve the goal.

In one experiment, participants performed the task for four days with different retention intervals (e.g., 7 day – 1 day – 1 day, 1 day – 1 day – 7 day, or 1 day – 1 day – 7 day). Anderson, Fincham, and Douglas (1999) applied the strength accumulation equation to predicting latency results. The strength accumulation equation is as follows.

$$Strength = \sum_{j=1}^{n} t_j^{-d} \qquad \textbf{Equation 2.5}$$

t_j: the time that has passed since the jth occurrence of the item

d: decay rate

The strength accumulation equation is a base for the ACT-R mechanism to represent latency. When a participant performs a task, latency is an inverse function of the strength equation. This latency function serves for both power law of learning and power law of forgetting at the same time. The latency function provided a good fit of the theory to the empirical data.

$$Latency = A + B / \sum_{j=1}^{n} t_j^{-d}$$ **Equation 2.6**

A: the asymptotic latency

B: the amount of the latency that can be reduced by practice.

2.6.2 Learning Models of Procedural Memory

Previous experience can affect the current behavior. There are models supporting the learning of previous experience. ACT-R 5 (Anderson & Lebiere, 1998) selects one production to fire among competing productions, to achieve the model's goal. The mechanism allows a model to learn problem-solving strategies from experience based on the probability of success and the relative cost of different strategies in a production. Each production rule is associated with a utility value indicating how much the production is able to achieve the model's current goal ($U_i = P_i G - C_i + \varepsilon$). P_i is the expected probability to successfully achieve the model's current goal. The probability is decomposed to q and r ($P = qr$, where q is the probability that a production will achieve its intended next state, and r is the probability that the production achieves its goal when it arrives at the intended next state). C_i is the expected cost to achieve the model's objective. G is the value of the goal. ε is noise.

The probability of success is calculated by the number of successes divided by the number of successes and failures, as shown in Equation 2.4. q is assumed to be 1.

$$P = r(t) = \frac{Successes(t)}{Successes(t) + Failures(t)}, \ q = 1$$ **Equation 2.7**

This is the probability learning equation in ACT-R 5. Lovett (1998 p. 265) proposed time-based decay in ACT-R's production parameter learning. This mechanism discounts past experience and adjusts the timing of successes and failures. Similar to the ACT-R's base level activation, each success and failure experience in a production is decayed in terms of a power function.

$$Successes(t) = \sum t^{-d} \quad \textbf{Equation 2.8}$$

$$Failures(t) = \sum t^{-d} \quad \textbf{Equation 2.9}$$

Lovett's model has successfully explained behavior representation of adaptive choice. However, the mechanism is restricted to learning to the binary feedback of success or failure, leading to whether a reward is received or not. Thus, it could be impossible to represent choice behavior that is sensitive to the probabilities and magnitudes of reward separately (Fu & Anderson, 2004, 2006).

To address this problem, Fu and Anderson (2006) tested the reinforcement learning mechanism to explain complex behavioral data of recurrent choice and skill learning in the ACT-R's production system to provide more flexible continuous values of reward. Reinforcement learning is where humans or other creatures learn from interaction with the environment in an attempt to maximize rewards while achieving a goal.

Fu and Anderson's reinforcement learning mechanism is based on a *temporal difference* algorithm (Sutton & Barto, 1998) that is a generalization of the Rescorla-Wagner learning rule (Rescorla & Wagner, 1972).

$$V_i(n) = V_i(n-1) + a[R_i(n) - V_i(n-1)] \quad \textbf{Equation 2.10}$$

$V_i(n)$: the value of an item i after its nth occurrence

$R_i(n)$: the reinforcement of a reward or a penalty received on the nth occurrence

a: the rate of learning, $0 < a < 1$

Equation 2.10 is known as the difference learning equation. One notable feature of the difference learning equation is that the model can represent the learning process in a continuous range instead of the "success" or "fail" information on the action (Fu & Anderson, 2006).

The difference learning equation is elaborated in the temporal difference algorithm by defining the immediate reward, $r_i(n)$, and the value of the next item $i+1$, $V_{i+1}(n-1)$, as follows.

$$R_i(n) = r_i(n) + g(t_i)V_{i+1}(n-1)$$ **Equation 2.11**

In Equation 2.11, the term of $g(t_i)$ indicates the discount function representing monotonic decrease in values with time, t_i. There are some arguments that what the discount function would be (e.g., exponential function). Fu and Anderson provided a good summary of this question and showed the useful incorporation of the discounting function in the ACT-R's learning mechanism.

Fu and Anderson designed a general skill learning task of the maze-searching that how humans learn with delayed feedback to test how well the reinforcement learning mechanism can scale up to account for skill learning. Participants (N = 20) were asked to make recurrent choices to progress through simulated rooms to reach a goal. In this task, the main component of the skill is to acknowledge when to apply the right choice under a given context of cues in the task environment. Fu and Anderson asserted that the learning of making right choices could be considered as a core component in skill acquisition.

They concluded that the reinforcement learning mechanism propagates discounted credit back to previous productions, leading to learning of rewards with psychological validity. That is, the credit assignment mechanism allows the model to learn the task and improve performance as correct production rules are chosen to fire.

This investigation provides a couple of benefits to modeling skill acquisition. First, this recurrent choice model can represent how a value is discounted with delay. Second, the model can represent how the reward value links back to the previous action over time. Also, the model with the conflict resolution equation can provide a stochastic-dynamic description of the recurrent choice of learning and performance.

2.6.3 Forgetting Models of Declarative Memory

In this section, I will review forgetting models of declarative memory rather than procedural memory. The models are based on activation and spreading mechanisms.

Anderson's Fan Effect

As humans acquire more facts, their time to retrieve a fact from memory increases. This phenomenon is called the *fan effect* (see Anderson, 1974; Anderson & Reder, 1999). The fan effect provides an understanding of retrieval processes that are interacting with human memory representations and a framework to explain associative strengths among acquired facts. Importantly, the study of the fan effect describes a theory of retrieval mechanisms that is a base of building a computational model in ACT-R.

Anderson (1974) conducted a set of experiments where participants learned 26 facts about people and locations, such as *A hippie is in the park*. After the learning sessions, participants were asked to determine whether sentences presented to them were true or false. The term of "fan" refers to the number facts associated with a particular concept. The study deals with how latency increases when the number of facts associated with a person or a location increases.

In the ACT-R theory, an activation value of a fact determines the latency to retrieve any fact from memory (Anderson & Reder, 1999). A fact would be comprised of chunks in declarative memory. The chunks spread activation based on their relation to other chunks, called their strength of association.

Knowledge and skills retention often adversely affect performance speed, such as response time. The functional relationship between spreading activation and retrieval latency can provide a clue to model skill decay. Chong (2004) also mentioned that the spreading activation through associative links between declarative elements can be applied to rules that generate decay.

Pavlik's Forgetting Model

Pavlik and Anderson (2005) studied the spacing effects and optimization of learning based on the ACT-R framework. The spacing effect is considered as the memory benefit that controlling the time duration between practices can enhance human performance.

They investigated the evidence of spacing effects through examining Japanese-English paired associates of declarative knowledge. From this vocabulary paired-associates experiment, an activation-based model was proposed. This model proposes that an item receives an increment of strength when it is practiced, but that these increments decay as a power function of time.

Pavlik's activation model purely focuses on declarative knowledge. Thus, it is necessary to prove that the activation mechanism can be applied to dealing with the aspects of procedural knowledge and its degradation. Also, the spreading activation mechanism is needed to be proved in the context of procedural knowledge degradation. The spreading activation mechanism indicates that a repeatedly used chunk spreads activation through its associative links between chunks in declarative memory.

The basic activation mechanism is as follows:

(a) Creation of memory element
(b) Initial activation
(c) The activation begins to decay as a function of time.
(d) If the activation falls below a retrieval threshold, the memory element cannot be retrieved.
(e) The memory is not available to satisfy the condition of a rule.
(f) Activation of a memory element is boosted by using or rehearsing the element.
(g) The decay process immediately resumes.

2.6.4 Forgetting Models of Procedural Memory

I have run across mostly models on declarative memory degradation (see Anderson & Reder, 1999; Pavlik Jr. & Anderson, 2005). Chong (2004) investigated a modeling consideration of procedural skill decay in the ACT-R architecture. Chong mentioned that the existing set of mechanisms from several architectures (e.g., refer to the below Table 2.4) could not afford modeling procedural skill decay.

Table 2.4. Modeling capabilities of skill decay in cognitive architectures.

Architecture	Capability
EPIC	It does not provide a rule learning mechanism. This indicates that the architecture is not able to model procedural skill learning.
ACT-R	Usually, the architecture's performance is limited to declarative knowledge learning and forgetting.
Soar	As a rule learning mechanism, chunking is used to model learning phenomena but not the decay of skill.
EASE	This is integrated hybrid architecture (Elements for ACT-R, Soar, EPIC). Modeling of skill decay was tried but it does not have stronger capability than its three original architectures.

For example, EPIC does not provide a rule learning mechanism. In Soar, as a rule learning mechanism, chunking is used to model learning but not skill degradation. ACT-R is limited to learning and forgetting of declarative knowledge. Thus, it is worth exploring and extending the existing architectural mechanisms to model procedural skill degradation.

Brannon (2001) attempted to provide a computational model of procedural knowledge degradation within the ACT-R 5 architecture. It has symbolic and subsymbolic levels supporting performance and knowledge dynamics. The symbolic level consists of the primary equation for expected gain. The equation to compute the expected gain is denoted by $E = PG - C$ (E : Expected gain; P : Probability of successful goal achievement; G : Value of current goal; C : Cost of achieving goal).

This equation provides calculation of the expected gain of each production and the production with the highest expected gain is to be fired. The subsymbolic level records event data associated with production firing and supports parameters in the expected gain equation. In this cognitive architecture, the aspects of procedural knowledge are represented by frequency with which the production cycle is executed.

Anderson and Lebiere (1998) mentioned that the more often productions are fired, the more efficiently knowledge will be retrieved because of accrual of production strength. This indicates the relationships between training and retrieval processes. It was positively concluded that the number of production cycle iterations affected the retrieval efficiency.

As mentioned before, forgetting can be explained by a psychological theory of “interference”. Also, “cue unavailability” can play a role to address the phenomena in the architecture using the production system. Chong (2004) stated that forgetting may be attributable to the inability to retrieve (match) a rule that has insufficient declarative cues. Chong proposed ACT-R’s base level learning (BLL) mechanisms to address recency and frequency effects of acquired procedural knowledge rules.

In ACT-R, BLL is to determine the activation of working memory elements. Thus, the activation plays a functional role of knowledge availability and its retrieval time. According to the study, Chong used four parameters to explain skill decay including base level constant (β), declarative decay rate (d), noise (ε), and retrieval threshold (:rt). Particularly, the base level activation (B_i) explains the frequency and recency of a chunk, but the spreading activation of ACT-R was not considered for simplification purpose in the study.

The activation decays logarithmically as a function of time and frequency of a chunk’s use. The activation is augmented whenever the chunk is retrieved. After that, decay processes occur. In addition, the retrieval threshold parameter determines that if the activation of a chunk becomes below the threshold, a chunk will not be retrieved.

To contemplate on the modeling capability of ACT-R, it is needed to consider other parameters. The parameters, in ACT-R, are the instantaneous activation noise (:ans) and the latency factor (:lf). The instantaneous activation noise parameter determines how fast retrieval probability changes. The latency factor determines the magnitude of the activation effects on latency.

Chong concluded that the BLL mechanism can produce the desired skill decay effects with some open issues. For instance, ACT-R provides the spreading activation equation shown in Equation 2.12.

$$A_i = B_i + \sum_k \sum_j W_{kj} S_{ji} + \varepsilon \qquad \textbf{Equation 2.12}$$

A_i: the activation of a chunk i

B_i: the base-level activation

W_{kj}: the amount of activation from source j in buffer k

S_{ji}: the strength of association between source j and a chunk i

ε: the noise value that consists of a permanent noise associated with each chunk and an instantaneous noise computed at the time of a retrieval request

Using the spreading activation mechanism can make it possible that a relearned chunk spreads its activation to its associated chunks, resulting in an activation increase during a retention interval. This activation mechanism can have an implication on refresher training. That is, a strategic relearning can produce increases in a chunk activation leading to a better performance on retrieval.

Apparently, it is true that the activation mechanism in ACT-R is designed to explain declarative memory elements rather than procedural memory elements (production rules). One thing I emphasize here is that each production rule can refer to declarative memory elements. The activation of those declarative elements interacts with procedural knowledge. This suggests that procedural knowledge can be primed. Conversely, the primed declarative memory elements necessarily for procedural knowledge can be deactivated, leading to unpriming the procedural knowledge as well.

2.7 Cognitive Architecture for Training and Education

The contemporary sciences and technology of modeling human behavior are considerably indebted to Allen Newell's scientific desires and pursuits. In 1987, Allen Newell delivered the William James Lecture Series at Harvard University. The record of the lecture was published as a book entitled *Unified Theories of Cognition* (Newell, 1990). In that book, Newell strived for a unified theory of cognition through the Soar architecture and brought pivotal research motivations and questions to model human behavior. Also, in Newell's last lecture recorded back in 1991, he raised an ultimate scientific question, *How can the human mind occur in the physical universe?* Here is the excerpt of his question (Newell, 1993).

> The question for me is, how can the human mind occur in the physical universe? We now know that the world is governed by physics. We now understand the way biology nestles comfortably within that. The issue is, how will the mind do that as well?

As a candidate of a unified theory of cognition, John Anderson and his colleagues have proposed the ACT-R architecture. ACT-R has been validated as a cognitive architecture to address human behavior and learning in various tasks (Pew & Mavor, 1998). While Soar is more oriented for Artificial Intelligence, ACT-R has been actively used to model human cognition and performance. Recently, Anderson (2007) gives his answers responding to the Newell's scientific inquiry with description of brain image maps to the architectural functions.

The purpose of the cognitive architectures (e.g., Soar and ACT-R) is to provide a framework to build a model that can represent human behavior. Ideally, the architectures not only support but also confine modeling capabilities to allow models that are cognitively plausible or to reject models that do not match possible human behavior (Taatgen, Lebiere, & Anderson, 2006).

I chose, in this dissertation study, the ACT-R architecture to explore and model skill retention of human behavior, because ACT-R provides a unified theory of cognition that can explain human behavior of learning. In particular, specific mechanisms of ACT-R are close to phenomena of learning and forgetting.

2.7.1 Overview of the ACT-R Architecture

This project uses ACT-R 6. The ACT-R 6 architecture is evolving. If you want the most recent one, you should check out ACT-R's official website[1], visit the annual ACT-R workshop, or attend conferences (e.g., the Cognitive Science Society conference or the International Conference on Cognitive Modeling) where the ACT-R tutorial session might be provided periodically. Also, you could attend the annual ACT-R summer school.

[1] act-r.psy.cmu.edu

Figure 2.4 shows a schematic of the ACT-R architecture. As an extension of the binary classification of memory (declarative memory and procedural memory), ACT-R 6 has several other modules that are all associated with the production system as a center. The modules include the Intentional Module (Goal Module), Declarative Module, Aural Module, Speech Module, Motor Module, Vision Module, and Imaginal Module. The production system communicates with each module via each corresponding buffer. That is, the central production system plays a role of generation and coordination of behavior in accordance with the functions of modules.

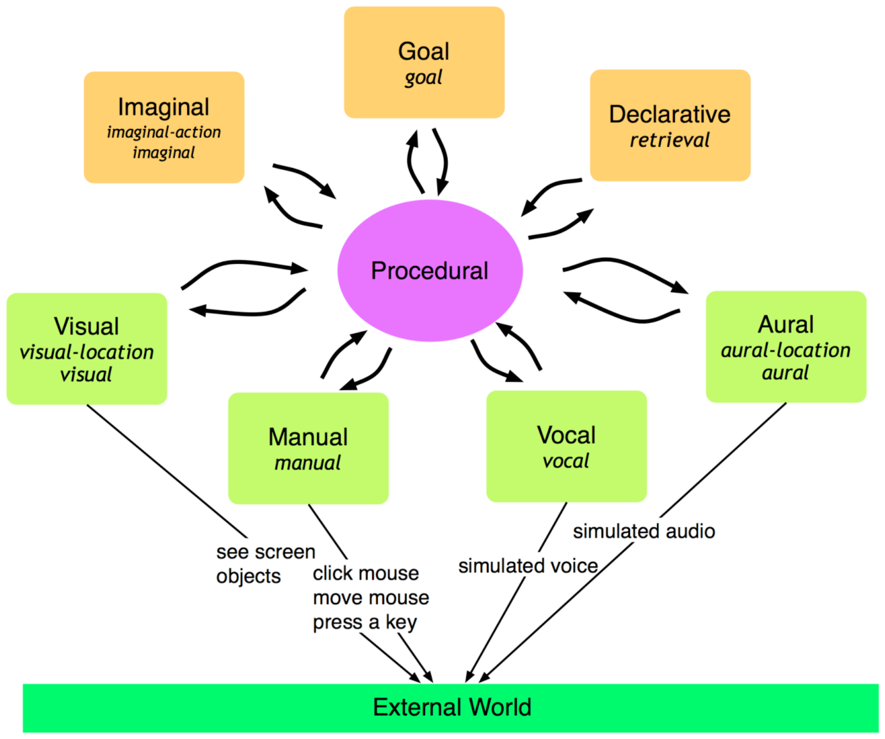


Figure 2.4. A schematic view of the ACT-R architecture, adapted from Anderson (2007) and Byrne (2001).

The buffers' roles (e.g., manual buffer, retrieval buffer, or goal buffer) are mainly two folded: (a) making a request from a production to a module, and (b) holding a chunk as a result of that request. Also, the buffers serve as the source of chunks for declarative memory. Here are the details of each module's roles:

The procedural module plays a central role in coordinating productions that interact with other modules. This module specifies productions and matches a production to fire with the utility module. The utility module provides quantitative values of the productions' subsymbolic utility used in the conflict resolution process. The other module is the production compilation module responsible for learning new productions. These three modules constitute the ACT-R's procedural system.

The goal module keeps track of current goals or intentions of the model. The goal buffer in this module tracks the current state of a model and holds relevant information for the current task. The only action of the goal module is to create new chunks and they are placed into the goal buffer.

The declarative module retrieves a chunk from memory. Declarative memory in this module stores chunks generated by the model. The retrieval of chunks reflects a theory of human memory performance.

The visual module identifies objects in the visual screen. ACT-R's vision module, consisting of the visual-location and visual buffer, provides the model with information about what it is and where it is on the screen. The module determines what the ACT-R model sees (Anderson et al., 2004; Byrne, 2001; Byrne & Anderson, 1998). Those "what" and "where" are the two subsystems in the vision module. If an object is on a visual display, it is represented by one or more features (e.g., location, color, or size, etc.) in the vision module's icon. Then, the module creates chunks based on the features providing declarative memory representations of the visual display. Chunks can then be matched by the condition side of productions and be placed in declarative memory after a successful match-up.

For the subsystem of "where", a request to identify what is at is sent to the vision module through the visual-location buffer. A modeler can specify x and y coordinates of the visual location (i.e., screen-x is greater than 120). If there is an object that meets the visual location specification, a chunk representing that object is placed into the visual-location buffer. If not, the buffer will remain empty.

For the subsystem of "what", a request on what is it is sent to the vision module through the visual buffer. The where system finds a specific location of an object, then the what system attends to that location by shifting visual attention and places a chunk representing that object in the visual buffer. One notable property of the where system is whether an object has been previously attended or has not been previously attended by the vision module. The ACT-R's vision module keeps track of a small number of locations that have been attended. Markers, called a finst (instantiation of finger), are used to track and are limited in both number and duration. The number of finsts are controlled by the command of :VISUAL-NUM-FINST, which defaults to 4. The duration, indicating how long a finst marker will remain in a location, is controlled by the command of :VISUAL-FINST-SPAN that defaults to 3.0 s. As another property, the ACT-R's vision module supports visual tracking. As a visual object is attended to, the vision module is able to track that object by a START-TRACKING operator (Byrne, 2001).

The motor module produces the hand movements such as a key press, mouse move, or mouse click. ACT-R assumes that the model's hand operates a virtual keyboard with a layout of keys in two dimensions and a virtual mouse. The virtual keyboard is shown in Figure 2.5.

For example, the key of "j" is represented by `(7 4)` in ACT-R. It is assumed that the virtual mouse is located in `(28 2)`. The mouse has only one button. The model's right hand controls the mouse. At default, the model's index fingers of the left and right hand are placed over the `F` and `J` keys on the keyboard respectively, which is at `(4 4)` for the left index finger and `(7 4)` for the right index finger.

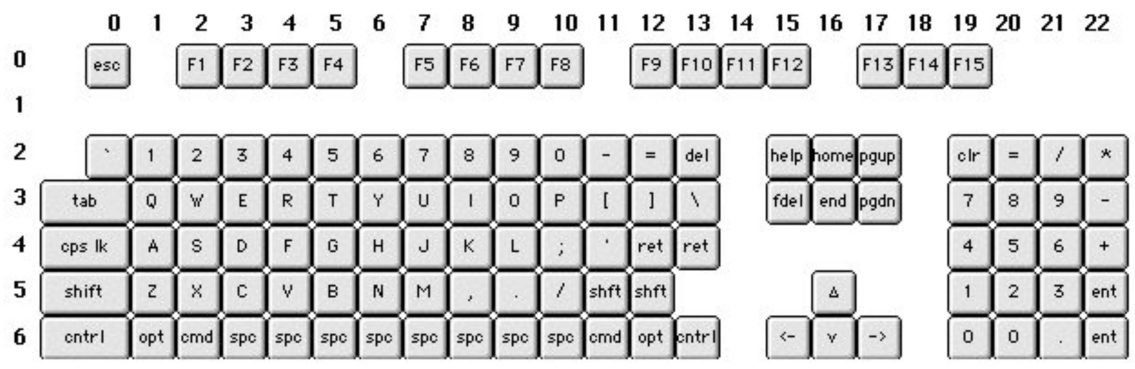

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
0 esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15
1
2 ` 1 2 3 4 5 6 7 8 9 0 - = del help home pgup clr = / *
3 tab Q W E R T Y U I O P [ ] \ fdel end pgdn 7 8 9 -
4 cps lk A S D F G H J K L ; ' ret ret 4 5 6 +
5 shift Z X C V B N M , . / shft shft ∆ 1 2 3 ent
6 cntrl opt cmd spc spc spc spc spc spc spc spc cmd opt cntrl <- v -> 0 0 . ent

Figure 2.5. The virtual keyboard in ACT-R.

The imaginal module has a buffer called imaginal. This buffer creates new chunks that are the model's internal representation of information. The imaginal buffer maintains context relevant to the current task. There is a different mechanism between the goal and imaginal buffer. In the goal buffer, chunks are created and placed before the start of the model. On the other hand, in the imaginal buffer, the imaginal module creates a chunk in the imaginal buffer by a production's request.

For example,

```
(p read-cell
   =goal>
      isa                read-cell
      state              attend
   =visual>
       isa              cell
       value            =number
==>
    =goal>
       state             respond
    +imaginal>
       isa               array
       cell-value        =number)
```

This production requests the imaginal module to create a chunk. It automatically clears the imaginal buffer, and then the new chunk is placed into the imaginal buffer. At default, it takes 0.2 seconds for that chunk to be available. ACT-R takes time to build a chunk representation and creates one new chunk at a time. The default value can be changed by a parameter. To sum up, after firing the production rule, the model reads from the spreadsheet screen, the information (e.g., a cell value) is placed into the imaginal buffer.

2.7.2 ACT-R Model: Lessons Learned For Training and Education

The engineering model by Card, Moran, and Newell (1983) made a significant contribution to research on cognitive modeling, particularly to human-computer interaction. Ritter et al. (2000) summarizes contributions to human computer interaction (HCI) in three main ways: (a) examine efficacy of different HCI designs by predicting task performance, (b) provide embedded intelligent assistants in education, and (c) create models of users that reside in synthetic environments for an advanced simulation (e.g., fighter pilot performance). The laborious work of cognitive modeling has produced those deliverables and contributions to HCI.

With the perspective of those, the work of cognitive modeling can be theoretically applied to training and education and practically provide implications to those, because cognitive modeling is grounded in theories of human cognition (see Gray & Altmann, 2001), providing meaningful implications for practical problems. When it comes to training and education, cognitive modeling can (a) examine efficacy of different training regimens, (b) provide cognitive tutors in training and education, and (c) create models of users in a synthetic environment.

First and foremost, modeling of human performance requires knowledge that represents human's goal directed behavior. Newell (1990) established important foundations to cognitive systems in his book, *Unified Theories of Cognition*. In that book, Newell mentioned *mind* and *response functions* to describe a system's behavior. *Mind* can be considered as a control system that directs behavior of the system complexly interacting with the dynamic world. The *mind* provides actions as a function of the external environment called *response functions*. Thus, behavior of the cognitive system changes by response functions to provide physical and biological nature of human beings.

Newell (1990) describes the cognitive system as a full range of behavior by claiming that humans can be represented at the knowledge level. The reason is that if we describe a system at the knowledge level, it is not always necessary to know the system's detailed internal processing. If we know the system's goal and what the system knows about its environment, behavior of the system can be calculated (Newell, 1990). This implicates that there should be a way of formulating humans as agents that have knowledge and goals. Based on foundational structures by Newell, Soar and ACT-R are two of the most commonly used cognitive architectures (Ritter et al., 2003). Both Soar and ACT-R use a production system to represent knowledge in memory.

It is necessary to consider whether the production system can afford theoretically meaningful and psychologically plausible human behavior. Generally, memory structures are comprised of binary distinction of declarative memory and procedural memory. In

Soar, the production system consists of production memory that is permanent, and working memory that is a collection of data elements (Newell, 1990). Learning in Soar occurs only for production memory—that is, new rules are created by the Soar architecture whenever a sub-goal is resolved by a chunking mechanism, and, unlikely, learning in ACT-R involves both declarative and procedural memory (Ritter et al., 2003). I put more focus on the ACT-R architecture across this dissertation study than Soar because ACT-R has more psychologically plausible memory constructs with binary distinction. This can be a crucial factor to understand human memory and learning performance.

The ACT-R architecture has three features for knowledge representation. First, ACT-R has two long-term repositories of knowledge, that is declarative and procedural memory, representing different memory processes. Second, a chunk represents knowledge in declarative memory, representing factual information. Third, a production represents the basic unit of the knowledge in procedural memory, representing a goal-directed behavior.

Anderson (1993) expounds the binary distinction of declarative and procedural memory within a production system framework where productions function by reading information from working memory and writing information to working memory. The information in working memory is declarative knowledge and the information in the productions is procedural knowledge. ACT-R supports active and inactive properties of information, whereas Soar loses information when that information leaves working memory. Thus, ACT-R allows long-term status of knowledge in memory. In ACT-R theory, declarative knowledge comes from encoding of the environment, and procedural knowledge must be compiled from declarative knowledge through practice. Also, the spreading activation implements declarative priming in ACT-R. Furthermore, declarative knowledge structure loses associative strength with time.

Anderson (1993) gave a good example of learning typing skills that helps us to understand ACT-R's binary distinction between declarative knowledge and procedural knowledge. One can memorize the layout of the keyboard declaratively, and one can know the keyboard as part of our typing skill. Learning typing skills enables one to memorize the keyboard layout and to type faster and faster. A typist declaratively memorizes the keyboard layout and procedurally knows the keyboard as part of the typing skill. Over time, several months or more, a typist can lose declarative knowledge of the keyboard layout but can retain typing skills. The way the typist can determine where a key is on the keyboard is to imagine typing a letter and seeing where his/her finger goes. This example indicates that one can and do maintain both declarative and procedural representations of the same knowledge.

Knowledge representation of using chunks has three major features in ACT-R. First, a size of chunk can be considered for modeling of recall performance. However, in general, model performance of skill acquisition is not sensitive to the size limitation of chunk (Anderson, 1993). Second, chunks can be encoded with respect to their semantic relationships. Thus, it is possible to represent specific relational roles in knowledge using elements of chunks. Third, in representational notation of knowledge, chunks can be hierarchically organized.

Cognitive skills are realized by production rules because production rules are the right grain size for understanding skill acquisition (Anderson, 1993). Knowledge representation of using productions that are the units of procedural knowledge has four major features in ACT-R as follows:

(a) Production Modularity: All productions are independent with each other, so that one can delete and add productions like a separate element. Each production rule ideally should represent the basic unit of skills based on the task analysis. Thus, it is said that skills are acquired in production-sized units, and any transformations of skills exist when each production unit changes (Anderson, 1993).
(b) Abstraction: Production abstraction refers to the generality of production rules. That is, production rules can be administered in terms of the pattern specification of the condition.
(c) Goal-structuring: The internal goal in the cognitive system can determine firing of different productions rules in response to the same external situation. This goal-structuring in ACT-R supports adaptiveness of the cognitive system.

(d) Condition-action asymmetry: The flow of control goes from the condition side to the action side. The reverse flow from action to condition is not possible in production rules.

2.8 Summary

In Chapter 2, I have discussed several important topics, including knowledge and skills, acquisition, degradation, and retention. Procedural knowledge and skills are ubiquitous in our daily tasks. An understanding of learning and forgetting procedural skills helps to devise design of experiments to measure learning and forgetting in a laboratory setting.

Thus, we need procedural task that should be relatively large to represent the real world task, instructions to have human participants learn the task, and multi-days of retention that can help to forget skills. These are the basic foundation to design experiments.

Also, I reviewed existing models that address learning and forgetting. The ACT-R cognitive architecture is chosen to develop a cognitive model because the architecture is close to explain phenomena of learning and forgetting. Based on the review, I will describe areas for exploration and the detailed research design in the next chapter.

Chapter 3
Areas for Exploration and the Research Design

This chapter describes the detailed research design and methods to explore learning and forgetting procedural skills. Enhancing skill retention (or reducing skill decay) has gained great interest in industry and military sectors, as I discussed in the previous chapter, but little research has been conducted to model skill degradation phenomena (Chong, 2004). It is worth developing a cognitive model producing human behavior of skill degradation and finding out how skills are acquired and retained.

As mentioned in Chapter 1, the objective of this study is to develop a cognitive model of skills decay to better summarize the training issues of knowledge retention and forgetting. My hypothesis of this study is that the ACT-R theory of the subsymbolic computation mechanisms (activation and production compilation) cannot capture skill decay performance.

To achieve the objective, the dissertation study is two folded. First, a computational model in a cognitive architecture (Chapter 4) is to be explored and implemented. Second, I create a study paradigm to explore and gather human data of skills degradation (Chapter 6). Finally, I will validate the model performance by comparing it with human data (Chapter 7).

3.1 The Environment to Explore Skills Degradation

Figure 3.1 shows a schematic view of dissertation research study. The left side of the figure is called the ESEGMAN world. This is where a cognitive model resides. The model is built within the ACT-R architecture that is written in Common Lisp. The model interacts with the target task environment. The target task is a set of spreadsheet tasks. The spreadsheet application, called Dismal, was written in Emacs Lisp (Ritter & Wood, 2005). Both the model and the spreadsheet application reside in the Emacs text editor.

To achieve the interaction between the model and the target task, it is necessary to implement a substrate between them. This is called ESEGMAN (Emacs Substrate: Gates toward MAN-made world), represented in an eye and a hand in the ESEGMAN World.

The ESEGMAN world is a simulated environment consisting of the cognitive model, ESEGMAN (a substrate), and the target tasks.

In the right side of the figure, human participants perform the same target task in the same Dismal spreadsheet application through the Emacs user interface. To analyze human behavior, RUI (Recording User Input) is used to record subjects' keystrokes and mouse movements (Kukreja, Stevenson, & Ritter, 2006).

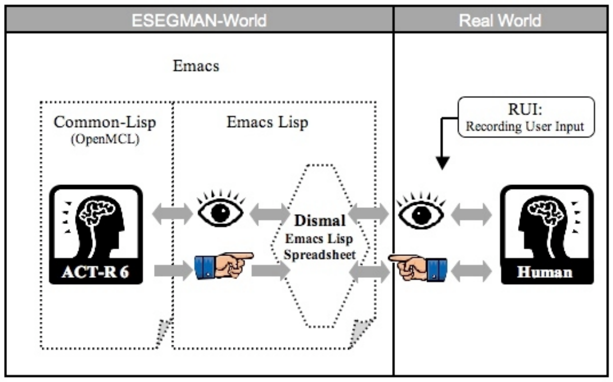


Figure 3.1. A Schematic representation of the study environment.

The Model

Pavlik and Anderson (2005) implemented a forgetting model but this model only addresses vocabulary memory of declarative knowledge. Chong (2004) studied an architectural way to model skill decay. He proposed the application of the activation mechanism to rules of procedural knowledge but could not provide the real task interactions with a cognitive architecture. In this study, a cognitive model in ACT-R is attempted to produce human behavior of learning and forgetting.

ESEGMAN

There are restrictions of the cognitive models with regard to accessing to the external man-made environment. ACT-R supports the perceptual and motor capability but it is not complete to interact with the environment.

ESEGMAN (Emacs Substrate: Gates toward MAN-made world) is implemented to overcome the technical challenges. ESEGMAN is a substrate to interface an ACT-R model with a man-made task environment. In the ESEGMAN world, Emacs acts as an operating system in which the ESSEGMAN embodies a cognitive model to interact with the actual task. This study opens a possibility of a new cognitive modeling paradigm and empowers ACT-R's vision and motor capabilities.

The Task

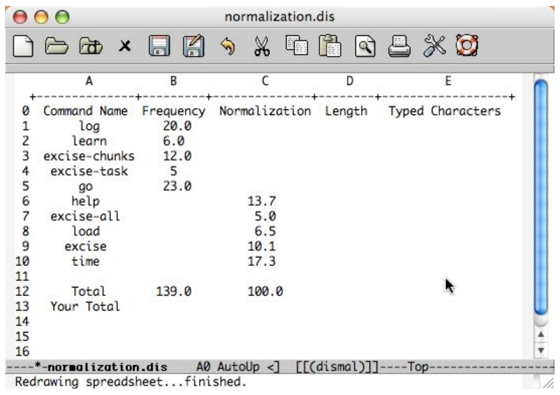
Card, Moran, and Newell (1983) studied how a user's skill would interact with computer-based systems with a focus on the text editing task domain. This research of text editing skills has provided important findings on human performance and information processing. Singley and Anderson (1989) investigated the transfer of cognitive skills in the text editing tasks by providing in-depth theory of learning through the ACT* architecture.

In this study, as an extension of text editing task, a set of spreadsheet tasks is used to measure procedural knowledge and skills degradation. The spreadsheet task is expected to provide knowledge and skills from perceptual-motor to the cognition-demanding task characteristics. Thus, the Dismal spreadsheet task, unlike previous research on text editing tasks, can give a balanced set of knowledge and skills including procedural cognitive, declarative, and perceptual-motor knowledge.

A spreadsheet called *Dismal* was implemented to gather and analyze behavioral data. Dismal extends the GNU Emacs editor using GNU Emacs' extension language, Emacs Lisp (Ritter & Wood, 2005). Dismal has three features of particular interest to those studying behavior: (a) the ability to manipulate and align sequential data, (b) an open architecture that allows users to expand it to meet their particular needs, and (c) an instrumented and accessible interface for studies of human-computer interaction (HCI). Figure 3.2 shows the Dismal spreadsheet in Emacs.

The Real World

In the real world, humans directly perform Dismal spreadsheet tasks. Human performance can be recorded by using RUI (Recording User Input) in the simulation environment (Kukreja, Stevenson, & Ritter, 2006). RUI records key presses, mouse moves (trace of locations in pixels), and mouse clicks of users in milliseconds. The RUI output of a text file can be incorporated to other statistical analysis package (e.g., the R language for computational statistics language) for analyses. Details on running experiments with human participants are described in Chapter 6.



```
           A            B            C          D            E
 +-------------+---------+-------------+--------+-----------------+
 0 Command Name  Frequency  Normalization  Length   Typed Characters
 1      log        20.0
 2     learn       6.0
 3 excise-chunks   12.0
 4  excise-task     5
 5      go         23.0
 6     help                     13.7
 7  excise-all                    5.0
 8     load                       6.5
 9    excise                    10.1
10     time                     17.3
11
12     Total      139.0        100.0
13  Your Total
14
15
16
----*-normalization.dis    A0 AutoUp <]   [[(dismal)]]----Top-----------------
Redrawing spreadsheet...finished.
```

Figure 3.2. The Dismal spreadsheet task window.

3.2 Task Analysis

I analyzed the Dismal spreadsheet task. The task analysis provides an understanding of decomposed components of the spreadsheet task. The components include attention shifts, encoding of information, attending to information, key presses, and mouse moves/clicks. The task analysis is a theoretical base to develop and test a computational model against complex and dynamic Dismal spreadsheet tasks.

3.2.1 Why Decompose the Task?

Lee and Anderson (2001) tested the reducibility hypothesis—complex tasks consist of lots of small components in an air traffic controller task. The learning of this complex task is decomposed into the learning of the small components. Lee and Anderson's decomposition of the task supported that the learning at the low level (smaller components) is consistent with the learning at the higher levels. This proves the reducibility hypothesis. At this point, it is necessary for us to consider the reducibility hypothesis is also applied to the forgetting process of humans.

Taatgen and Lee (2003) studied how to bridge a gap between learning simple tasks and performing complex tasks by investigating a skill acquisition mechanism called production compilation with which a computational model of a complex task was implemented.

This task analysis of hierarchical decomposition is based on Card, Moran, and Newell's (1983) task analysis—a task is decomposed into four levels (e.g., the unit-task level, the functional level, the argument level, and the keystroke level). Lee and Anderson (2001) left out decomposition at the argument level in their study because it consists of instantiations of the functional level goals. I will follow Lee and Anderson's method to analyze the Dismal spreadsheet task because they successfully extended Card et al.'s task analysis on expert performance to the analysis of learning at the following three levels:

- Unit task level: The goal at the unit task level is repeatedly executed to achieve the main task goal.
- Functional level: At this level, the unit task goals are further decomposed into smaller and functional level goals.
- Keystroke level: This is the most detailed level consisting of elementary motor and cognitive goals. The goals at this level include goals of key-press, encoding of the information in the environment, and information retrieval from long term memory.

3.2.2 The Dismal Spreadsheet Task

Two input modalities were used for the task. Some participants use a keyboard with key-based commands. Other participants use a vertical mouse with menu-based commands. A vertical mouse is chosen, shown in Figure 3.3, because it provides new motor skills to learn and forget. This vertical mouse requires different hand and forearm postures, instead of a palm-down position of a regular mouse. It is ergonomically designed to reduce stress on a user's wrist. None of the participants had prior experience of using this vertical mouse so we could minimize participants' previous knowledge and skills.

Figure 3.3. A vertical mouse (Evoluent™ VerticalMouse).

Participants performed a set of novel spreadsheet tasks. The spreadsheet task had 14 steps as follows:

(1) Open a file, named normalization.dis under the "experiment" folder
(2) Save as the file with your initials
(3) Calculate and fill in the frequency column (B6 to B10)
(4) Calculate the total frequency in B13
(5) Calculate and fill in the normalization column (C1 to C5)
(6) Calculate the total normalization in C13
(7) Calculate the length column (D1 to D10)
(8) Calculate the total of the "Length" column in D13
(9) Calculate the Typed Characters column (E1 to E10)
(10) Calculate the total of the "Typed Characters" column in E13
(11) Insert two rows at A0 cell
(12) Type in your name in A0
(13) Fill in the current date in A1 using the command
(14) Save your work as a printable format

The A column ("Command Name")—as shown in Figure 3.2, has ten different names of computer commands (A1 to A10). The B column ("Frequency") has frequencies of each command listed in the A column. There are default values of each frequency (B1 to B5). Participants calculate frequencies of each command from B6 to B10 using Equation 3.1. Normalized frequencies are listed in the C column ("Normalization"). While the cells (C6 to C10) are of default values of normalized frequencies, participants need to calculate the blank cells of C1 to C5, using Equation 3.2. In the D column ("Length"), participants need to calculate the length of each command by using a Lisp function of `length`. The typed characters in the E column are calculated by multiplying a command name's frequency by its length. The totals of each column

(B13, C13, D13, and E13) are to be calculated. Then, participants need to insert two rows at the first row and type a participant's name and the current date by using a Dismal command, `(dis-current-date)`. Finally, the last step is to save the work as a printable format.

$$\text{Normalization} = \frac{(\text{Frequency} \times 100.0)}{\text{Total frequency}} \qquad \textbf{Equation 3.1}$$

$$\text{Frequency} = \frac{(\text{Normalization} \times \text{Total freqeuency})}{100.0} \qquad \textbf{Equation 3.2}$$

The Dismal spreadsheet task offers two different input modalities: keyboard and mouse. Each modality characterizes the task by different types of knowledge and skills.

Keyboard Users

Keyboard users are only allowed to use key-based commands. They are trained not to use the menu bar with a mouse. For example, when they open a file they have to use a key-based command of C-x C-f. Also, when moving around the cells in the spreadsheet, they are only allowed to use corresponding key-based commands (C-f for moving to right, C-b for moving to left, C-p for moving up, and C-n for moving down).

Mouse Users

Participants using a vertical mouse are only allowed to use menu-based commands when they open a file or save the file as another name. When moving around the cells, they are only allowed to use the mouse. Table 3.1 and Table 3.2 show the task analysis with all subtasks for keyboard and mouse users.

Table 3.1. Subtasks of the Dismal spreadsheet task for keyboard users.

Subtasks	Keystrokes	Mouse actions
(1) OPEN FILE	Press `C-x C-f` Type in normalization.dis ↲	N/A
(2) SAVE AS	Press C-x C-w Type in JWK.dis ↲	N/A
(3) CALCULATE FREQUENCY (B6 to B10)	Move the point to B6 by using C-p, C-n, C-f, or C-b Press `e` Type in the frequency equation of `(/ (* c6 b12) 100.0)` ↲ Repeat for B7 to B10	N/A
(4) CALCULATE TOTAL FREQUENCY (B13)	Move to the point to B13 Press **`e`** Type in the formula of `(dis-sum b1:b10)` ↲	N/A
(5) CALCULATE NORMALIZATION (C1 to C5)	Move the point to C1 Press `e` Type in the normalization equation of `(/ (* 100.0 b1) b12)`↲ Repeat for C2 to C5	N/A
(6) CALCULATE TOTAL NORMALIZATION (C13)	Move the point to B13 Press **`e`** Type in the formula of `(dis-sum c1:c10)` ↲	N/A
(7) CALCULATE LENGTH (D1 to D10)	Move to the point D1 Press `e` Type in the formula of `(length a1)` ↲ Repeat for D2 to D10	N/A
(8) CALCULATE TOTAL LENGTH (D13)	Move the point to D13 Press `e` Type in the formula of `(dis-sum d1:d10)` ↲	N/A
(9) CALCULATE TYPED CHARACTERS (E1 to E10)	Move the point to E1 Press `e` Type in the formula of `(* b1 d1)` ↲ Repeat for E2 to E10	N/A
(10) CALCULATE TOTAL TYPED CHARACTERS (E13)	Move the point to E13 Press `e` Type in the formula of `(dis-sum e1:e10)` ↲	N/A
(11) INSERT TWO ROWS	Move the point to A0 Press C-u 2 i r ↲	N/A
(12) TYPE IN *NAME* (A0)	Press `e` Type in *`Name`* ↲	N/A
(13) INSERT CURRENT DATE (A1)	Move the point to A1 Press `e` Type in the formula of `(dis-current-date)` ↲	N/A
(14) SAVE AS …	Press `C-x C-w` Type in normalization-initials.dp ↲	N/A

Note: Information on subtasks including what keystrokes and mouse actions occur. The symbol of ↲ indicates pressing the carriage return key.

Table 3.2. Subtasks of the Dismal spreadsheet task for mouse users.

Subtasks	Keystrokes	Mouse actions
(1) OPEN FILE	N/A	Go to `File`>`Open File`> `experiment`>`normalization.dis`↲
(2) SAVE AS	Type `file-name with initials`↲	Go to `dFile` Select `Save buffer as...`
(3) CALCULATE FREQUENCY (B6 to B10)	Type in `(/ (* C6 B12) 100.0)`↲[Repeat for B7 to B10]	Move to and select B6 Go to `dEdit` Select `Edit cell (E)`
(4) CALCULATE TOTAL FREQUENCY (B13)	Type in `(dis-sum B1:B10)`↲	Move to and select B13 Go to `dEdit` Select Edit cell
(5) CALCULATE NORMALIZATION (C1 to C5)	Type in `(/ (* 100.0 b1) b12)`↲ [Repeat for C2 to C5]	Move to and select C1 Go to `dEdit` Select `Edit cell (E)`
(6) CALCULATE TOTAL NORMALIZATION (C13)	Type in `(dis-sum C1:C10)`↲	Move to and select C13 Go to `dEdit` Select `Edit cell (E)`
(7) CALCULATE LENGTH (D1 to D10)	Type in `(length A1)`↲[Repeat]	Move to and select D1 Go to `dEdit` Select `Edit cell (E)`
(8) CALCULATE TOTAL LENGTH (D13)	Type in `(dis-sum D1:D10)`↲	Move to and select D13 Go to `dEdit` Select `Edit cell (E)`
(9) CALCULATE TYPED CHARACTERS (E1 to E10)	Type in `(* B1 B1)`↲ [Repeat]	Move to and select E1 Go to `dEdit` Select `Edit cell (E)`
(10) CALCULATE TOTAL TYPED CHAR. (E13)	Type in `(dis-sum E1:E10)`↲	Move to and select E13 Go to `dEdit` Select `Edit cell (E)`
(11) INSERT TWO ROWS	N/A	Move to and select A0 Select `dEdit` Go to `Insert` Select `row` [Repeat one more time]
(12) TYPE IN *NAME* (A0)	Type in *`Name`* ↲	Move to and select A0 Go to `dEdit` Select `Edit cell (E)`
(13) INSERT CURRENT DATE (A1)	Type `(dis-current-date)`↲	Move to and select A1 Go to `dEdit` Select `Edit cell (E)`
(14) SAVE AS …	Type in `normalization-initials.dp` ↲	Go to `dFile` Select `Save buffer as...`

Note: Information on subtask including what keystrokes and mouse actions occur. The symbol of ↲ indicates pressing the carriage return key.

3.2.3 Decomposing the Dismal Spreadsheet Task

The Dismal spreadsheet task can be composed into the following subtasks with the task goal of completing the Dismal spreadsheet task.

- **Unit Task Level**
  File Handling: (a), (b), and (n)
  Frequency Calculation: (c) and (d)
  Normalization Calculation: (e) and (f)
  Length Calculation: (g) and (h)
  Typed Characters Calculation: (i) and (j)
  Editing the Spreadsheet: (k), (l), and (m)

- **Functional Level** (This level includes the fourteen steps of procedures)
  Open a file, named normalization.dis under the "experiment" folder
  Save as the file with your initials
  Calculate and fill in the frequency column (B6 to B10)
  Calculate the total frequency in B13
  Calculate and fill in the normalization column (C1 to C5)
  Calculate the total normalization in C13
  Calculate the length column (D1 to D10)
  Calculate the total of the "Length" column in D13
  Calculate the Typed Characters column (E1 to E10)
  Calculate the total of the "Typed Characters" column in E13
  Insert two rows at A0 cell
  Type in your name in A0
  Fill in the current date in A1 using the command
  Save your work as a printable format

- **Keystroke Level**
  Attend the spreadsheet
  Move attention (up, down, right, and left)

Press a key (use both hands)

Press a mouse button

Move a mouse

Keystroke-Level Model for the Dismal Spreadsheet tasks

Table 3.3 describes quantitative comparisons of the Dismal tasks for the two modalities (menu-based commands using a vertical mouse and key-based commands using a keyboard). Like the assumption of the Keystroke-Level Model (Card et al. 1983), I assumed error-free expert behavior of the spreadsheet task in this comparison, such as pressing keys or positioning a cursor, etc.

Keyboard users use only a keyboard. They need to learn key-based commands (e.g., how to open a file, how to save, how to edit a cell, and how to move around cells). Mouse users use the vertical mouse and the same keyboard as the keyboard users. Mouse users do not learn the aforementioned key-based commands, and they use only the menu-based commands. Both keyboard and mouse users must learn commands for calculating frequency, normalization, length, and operands for the spreadsheet manipulation.

As a simple way to compare the execution time of the two modalities, we used the Keystroke-Level Model (KLM, Card et al., 1983). The model includes primitive physical-motor operators (K – keystroke, P – pointing, H – homing, and D – drawing), a mental operator (M), and a system response operator (R), as shown in Equation 3.3. Using the equation, the execution time is calculated, as shown in Table 3.3.

$$T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R \qquad \textbf{Equation 3.3}$$

In the interest of simplicity, we ignored the timing term of T_D (drawing lines) and T_R (system response). It is assumed that the mouse positioning time (T_P) is the average time of 1.1 s. The homing time (T_H), which is about hand movement between different physical devices, is the average of 0.4 s.

Users spend time on mental preparation while performing the task. For example, users mentally prepare what to press, what to retrieve from memory, or what to do for the next step. For each mental operator, the preparation time (T_M) is assumed to be 1.35 s.

In counting the number of keystrokes, shift or control keys are counted as a separated keystroke operation. The keystroke time for "an average non-secretary typist" (40 wpm) is 0.28 s, the keystroke time for "typing complex codes" is 0.75 s, and the keystroke time for "typing random letters" is 0.50 s (see Card, Moran, & Newell, 1983 p.264). Here, I calculated the average typing speed from 11 participants. The average typing speed was 0.60 s. As compared to Card, Moran, and Newell's data, the participants' typing speed was faster than the speed from typing complex codes. The typing speed of participants in this study is between the typing speed of "typing random letters" and "typing complex codes" by Card, Moran, and Newell.

Table 3.3. Task comparison of mouse and keyboard users.

	Mouse users	Keyboard users
Keystrokes	730	1,030
Mouse move	125	N/A
Menu Selection	87	N/A
Others	38	N/A
Mouse clicks	125	N/A
Menu selection	87	N/A
Others	38	N/A

For mental preparation, I considered a rule that each subtask should include one mental preparation. Furthermore, if a subtask contains a number of repetitions, I counted the number as the one for mental preparation. For example, the subtask for mouse users, OPEN FILE, can be denoted by M K K K K K P P P P P, indicating a mental preparation for the subtask, 5 times of keystrokes (pressing a button), and 5 times of positioning. I assumed that a mouse user's hand location is initially on the mouse. Thus, I counted it as 0. As another example, the subtask for keyboard users, CALCULATE FREQUENCY, consists of 5 times of mental preparation and 162 keystrokes. It is assumed that a keyboard user's hand is initially located on the keyboard. More detailed information on how I calculated the task execution time can be found in Table 3.4.

Table 3.4. Task execution time for the Dismal spreadsheet task (mouse and keyboard users), based on the Keystroke Level Model analysis.

		Mouse Users		
	M	K	P	H
Parameter	1.35	0.6	1.1	0.4
SUB1	1	5	5	0
SUB2	1	23	5	1
SUB3	5	150	15	9
SUB4	1	25	3	2
SUB5	5	164	16	10
SUB6	1	27	3	2
SUB7	10	223	30	20
SUB8	1	23	3	2
SUB9	10	165	30	20
SUB10	1	27	3	2
SUB11	1	5	5	1
SUB12	1	10	2	1
SUB13	1	30	3	2
SUB14	1	27	3	3
SUM	40	904	126	75
TIME	54.0	542.4	138.6	30.0
			TOTAL TIME	765.0

		Keyboard Users		
	M	K	P	H
Parameter	1.35	0.6	1.1	0.4
SUB1	1	33	N/A	N/A
SUB2	1	26	N/A	N/A
SUB3	5	162	N/A	N/A
SUB4	1	27	N/A	N/A
SUB5	5	173	N/A	N/A
SUB6	1	40	N/A	N/A
SUB7	10	215	N/A	N/A
SUB8	1	27	N/A	N/A
SUB9	10	194	N/A	N/A
SUB10	1	29	N/A	N/A
SUB11	1	39	N/A	N/A
SUB12	1	12	N/A	N/A
SUB13	1	28	N/A	N/A
SUB14	1	25	N/A	N/A
SUM	40	1030		
TIME	54.0	618.0		
			TOTAL TIME	672.0

Note: The parameters are used to calculate the execution time of the Keystroke-Level Model. The unit of time is in seconds.

As shown in Table 3.4, the task execution time for mouse users is 765.0 s and the execution time for keyboard users is 672.0 s. Mouse users take 95 s more time than keyboard users based on the Keystroke-Level Model. This analysis provides us with a quantitative comparison of two tasks. If the design preference is seeking faster performance, using a keyboard with key-based commands can be appropriate, challenging the arguments of the GUI efficacy. However, there is a limitation of this KLM analysis. That is, we do not predict any user performance on learning and forgetting, which is critical information in designing and planning training regimens.

3.3 A Study with Model Subjects: Implementing an ACT-R Model

As mentioned before, the purpose of the cognitive model is to predict procedural skill decay. There are some identified technical challenges. First, there are no such models of skill decay. Based on the activation mechanisms of ACT-R, it is challenged to apply the activation mechanism of chunks in declarative memory to controlling rules in production systems. Second, ACT-R uses a virtual keyboard that assumes the keyboard to be a two-dimensional array of keys. The problem is that ACT-R currently can only perform one-key press down. ACT-R cannot perform two-key press down at the same time.

For example, using a Dismal spreadsheet task, the model attempts to open a file, it needs to press “C-x C-f” (C stands for the control key). Pressing the control key and “x” key at once is not supported. This technical problem needs to be resolved. There are several tasks to achieve the first goal of implementing a cognitive model:

- **Task 1:** Build the ACT-R model.
- **Task 2:** Apply ACT-R’s learning mechanisms to model skills decay.
- **Task 3:** Complete ESEGMAN.
- **Task 4:** Evaluate the model against human performance data.

3.4 A Study with Human Subjects: Exploring Skills Degradation

The study with human subjects measures participants’ learning and forgetting of spreadsheet tasks. The purpose of this study is to investigate how procedural skills are learned in human memory. The findings will help us to understand how skills are learned and how to train them.

For clarification, terms used in this study are specified here.

- Study session: An experimenter will train participants using the Study Booklet (Users Guide for the Dismal Spreadsheet). Each study session is limited to a maximum of 30 minutes.
- Test session: Human participants will perform the given tasks. The task completion time will be measured. Keystrokes and mouse actions (clicks or movements) will be recorded by RUI, Recording User Input (Kukreja, Stevenson, & Ritter, 2006). This is denoted by O_i (observation).
- Retention interval: This indicates time period of skill disuse. After the last study and test session, a six-day, twelve-day, or eighteen-day retention interval will be randomly assigned to participants. Then, retained skills will be measured through a test session. This is denoted by X_i.

Development of Hypothesis

The research hypotheses are to test the validity of the activation and production compilation mechanisms in the ACT-R architecture. Testing hypotheses will provide a better understanding and validity of the ACT-R modeling approach.

- **Hypothesis 1:** The production compilation mechanism in ACT-R cannot support learning and forgetting of procedural skills.
- **Hypothesis 2:** The activation mechanism in ACT-R cannot afford learning and forgetting of procedural skills.

This study, based on the hypotheses, will explore the activation mechanism in the ACT-R architecture in an attempt to prove the validity of a model for procedural knowledge decay. This will provide the base for creating an understanding of mitigating factors against knowledge and skills decay.

Design of Experiment

To investigate learning and forgetting by different knowledge and skill attributes, the experiment is basically 2×3 design with 6 conditions. Two factors of skill types (learning modalities) are tripled by the variable of three different retention intervals. For the factors of skill types, the two levels include procedural cognitive tasks using a keyboard and procedural cognitive tasks using a vertical mouse.

The experimental design consists of two parts: (a) measure of learning; and (b) measure of forgetting in a time-series manner. There are study and test sessions for four consecutive days for learning. In a study session, participants are trained to complete the given task and they are tested to measure learning performance. Then, after three different retention intervals ranging from 6 to 18 days, forgetting will be measured through a test session.

Measure of Learning: Based on the pilot study, the study session will be less than 30-min and the test session will be less than 20-min. For the study session, participants will study a Study Booklet (Users Guide for the Dismal Spreadsheet) for a given time.

Measure of Forgetting: Participants will be asked to come back to the second, third, and fourth week to measure their forgetting on acquired spreadsheet skills. Basically, a participant's forgetting will be measured three times with a six-day retention interval. I have run a pilot study with one subject. Figure 3.4 shows the learning and forgetting curve on the Dismal spreadsheet task that is procedural and cognitive skills. The subject used a keyboard during the performance.

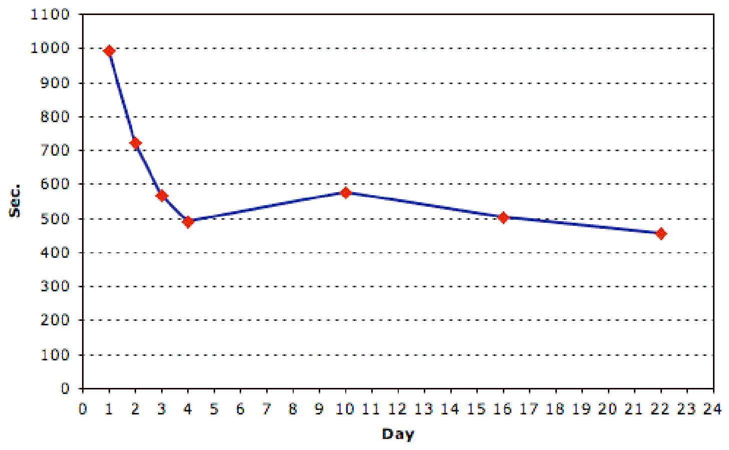


Figure 3.4. Learning and forgetting curve on the Dismal spreadsheet tasks (*N* = 1).

Table 3.5. Design of three different retention intervals for forgetting measures.

Subgroup 1	$O_1\,O_2\,O_3\,O_4$	$X_1\,O_5$	$X_1\,O_6$	$X_1\,O_7$
Subgroup 2	$O_1\,O_2\,O_3\,O_4$		$X_2\,O_6$	$X_1\,O_7$
Subgroup 3	$O_1\,O_2\,O_3\,O_4$			$X_3\,O_7$

I will move on toward describing detailed investigations based on the research design. I will start from exploration of the ACT-R theory and the skill retention model. Also, in a later chapter, I will report the analysis of the human data.

Chapter 4
The ACT-R Model and Mechanisms

This chapter addresses concise features of the ACT-R architecture where the Skill Retention Model is to implement. Architectural mechanisms are discussed to model skill learning and forgetting.

4.1 ACT-R's Symbolic Constructs

ACT-R is a hybrid cognitive architecture—it has symbolic and subsymbolic constructs. The symbolic construct is represented by ACT-R's production system. The subsymbolic construct is represented by a set of parallel processes in terms of a number of ACT-R equations that control symbolic processes.

4.1.1 The Production System in ACT-R

Cognitive psychologists or scientists have actively embraced the production system for computational modeling of human cognition and problem-solving (i.e., Anderson, 1976, 1982; Newell & Simon, 1972). Production systems support rules to represent behavioristic stimulus-response models and information processing—rules in the left-hand side can represent a stimulus or allow symbol processing of memories, goal, or plans, and rules in the right-hand side represent a response sequence, and then rules serve as units of behavior in terms of reaction time (Brownston, Farrell, Kant, & Martin, 1985).

In general, the basic structure of the production system consists of data memory (or working memory), production memory, and an inference engine (Brownston, Farrell, Kant, & Martin, 1985). *Data memory* functions as a global database of symbolic data items to represent facts related to the application domain or goals for a problem-solving strategy. Rules consisting of the condition and action part are stored in *production memory*. The condition part in productions describes configurations of elements in working memory and the action part details modifications to the working memory contents (Neches, Langley, & Klahr, 1987). An inference engine controls execution (or firing) of rules. The inference engine determines what to choose among rules based on data memory configuration. This selection process is called conflict resolution. Figure 4.1 shows a schematic view of a production system.

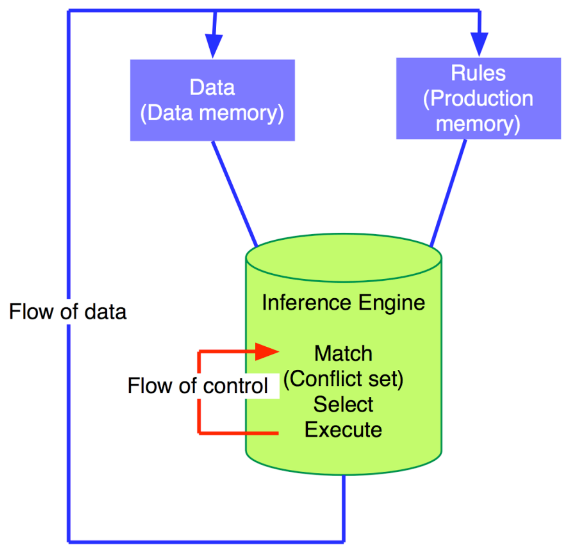


Figure 4.1. A general architecture of a production system.

Two interacting data structures including data (or working) memory and production memory are connected through a processing cycle called “recognize-act” cycle (Neches, Langley, & Klahr, 1987). The recognize-act process has three distinct stages: the match process, the conflict resolution process, and the act process. The recognize-act process operates in cyclic manner. From the basic idea of production systems, it can be said that the cyclic operations continue until no rules are matched or a stop command is encountered.

The behavioral repertoire of production systems can be changed by affecting the outcome from three distinct phases of the aforementioned “recognize-act” cycles including the process of matching productions, the process of conflict resolution, and the process of applying productions (Neches, Langley, & Klahr, 1987). In the process of matching productions, it is possible to consider two mechanisms such as generalization and discrimination. After finding a set of matching rule instantiations, the production system is, then, to make decisions on which instantiation in the set will be executed.

Thus, the conflict resolution process is to afford the selections of one or more of the instantiated productions.

There have been various theoretical research endeavors about the production system. Newell and Simon (1972) have proposed a formalism of a production system to build a model for problem solving. Anderson (1982) has used production systems in understanding skill learning processes. Also, Kieras and Polson (1985) stated that the production system formalism is directly related to the GOMS model, which was first proposed by Card, Moran, and Newell in 1983. For instance, “goals” in GOMS model appear in the condition statement in production rules and the action statement is to manipulate them. NGOMSL, an extension of GOMS reflects what humans do during their performance of procedural knowledge. NGOMSL which stands for “Natural GOMS Language” predicts learning and execution time on the basis of program-like representation of the procedures (Kieras, 1997). An important property of NGOMSL is based on production rule models. NGOMSL is a structured natural language to represent the user’s methods and selection rules. If methods are assumed to be of sequential and hierarchical form, NGOMSL can afford an explicit representation of the user’s methods.

As seen in Figure 4.1, the production system in ACT-R consists of production rules that can operate on facts in the declarative data memory. As mentioned in Chapter 2, this property allows ACT-R to have a distinctive specification of binary memory (i.e., declarative and procedural memory). Each production addresses the cognitive steps that are taken in performing a task (Anderson, 1982). ACT-R sequentially administers production rules corresponding to those cognitive steps to represent human cognition. When firing rules, the clauses in a production’s condition part must be matched against information that is active in data (or working) memory.

4.1.2 Flow of Control in Production Systems

One might wonder how production rules can be constructed to control sequences of behavior. In general, coding of production rules differ from other coding of conventional programming languages such as Java, C, or Fortran. Those programming languages provide commands for conditionals, loops, or recursions. Unlike them, the production system behaves differently and its flow of control does not depend on any explicitly programmed sequence but on recognition of ever-changing patterns in working memory (Brownston, Farrell, Kant, & Martin, 1985).

The inference engine in a production system performs a cyclic repetition with three states: (a) match, (b) select, and (c) execute, as shown in Figure 4.1. In the *match* state, the production system looks for all of the rules that are satisfied by the current contents in working memory. A collection of all of the candidate rules is referred to as the conflict set. In the *select* state, the system determines a rule in terms of a selection strategy. Then, the selected rule is executed, and the production system cycles back to the first state. A cycle of a rule-firing changes working memory. This cyclic control mechanism is called *recognize-act cycle*. Iteration of rules can be easy to program in the production system, because the *recognize-act cycle* is fundamentally similar to *do-while* loop (Brownston, Farrell, Kant, & Martin, 1985).

In ACT-R, productions are selected to fire via a process of *conflict resolution*. The *conflict resolution* decides one production to fire that matches the current goal. The ACT-R's symbolic structure supports interactions between chunks and productions in discrete cycles. In the meantime, the ACT-R's subsymbolic constructs with equations (e.g., activation-based computations) quantitatively determine qualitative properties of symbolic cognitive elements (Anderson & Lebiere, 1998).

4.1.3 Basic Production Patterns in ACT-R

In this section, I describe general patterns of production rules in terms of various ACT-R buffers. Example patterns are collected from existing models (i.e., models in the ACT-R tutorial and other models available). The total number of models I refereed to is fifteen models, and the number of total production rules is 132. Understanding the patterns can be helpful for modelers to build their own psychologically plausible models. Table 4.1 shows a set of brief summary of patterns in the ACT-R production system that I have frequently encountered[2].

[2] This summary of pattern is included in the ACT-R FAQ website, http://ritter.ist.psu.edu/act-r-faq/act-r-faq.html. If you find any frequent patterns and want to add to the FAQ, please email me, jongkim@psu.edu.

Table 4.1. A summary of production rule patterns in ACT-R.

Condition						Action					
g	**r**	**vl**	**v**	**i**	**m**	**g**	**r**	**vl**	**v**	**i**	**m**
=g	=r					=g	+r				
=g						=g	+r				
=g						-g					
=g	=r					=g					
=g	?r					=g					
=g			=v			=g				+i	
=g			=v	?i		=g				+i	
=g			=v	=i		=g				=i	
=g						=g		+vl			
=g		=vl	?v			=g			+v		
=g				=i	?m	=g					+m
		?vl						+vl		+i	
		=vl	?v						+v		
		?vl									

Note: g indicates goal buffer, r indicates retrieval buffer, vl indicates visual location buffer, v indicates visual buffer, i indicates imaginal buffer, and m indicates manual buffer.

Pattern 1	=g		=r			➔	=g		+r			

If the slot values in the goal buffer and the retrieval buffer are matched, then the production rule changes the goal and requests a retrieval from declarative memory. For example,

```
(P counting-example
   =goal>
      isa         count
      state       incrementing
      number      =num1
   =retrieval>
      isa         count-order
      first       =num1
      second      =num2
==>
   =goal>
      number      =num2
   +retrieval>
      isa         count-order
      first       =num2)
```

Pattern 2	=g					➔	=g		+r		

The condition part of the production tests the goal buffer, and then the action part changes the slot value in the goal buffer and requests a retrieval of a chunk from declarative memory. For example,

```
(P start
   =goal>
      isa         count-from
      start       =num1
      count       nil
==>
   =goal>
      count       =num1
   +retrieval>
      isa         count-order
      first       =num1)
```

Pattern 3	=g					➔	–g				

If the slot values in the goal buffer are matched, then the right hand-side of the production rule clears the goal by using the "-" operator.

For example,

```
(P stop
   =goal>
      isa         count-from
      count       =num
      end         =num
==>
   -goal> ; clear the chunk from the goal buffer
   !output!       (=num) ; print out the current number)
```

Pattern 4	=g		=r			➔	=g				

The left side of the production specifies values in the slots. In this condition part, if the goal is matched and a chunk is retrieved, then, the right side of the production modifies the goal. For example,

```
(P direct-verify
   =goal>
      ISA         is-member
      object      =obj
      category    =cat
      judgment    pending
   =retrieval>
      ISA         property
      object      =obj
      attribute   category
      value       =cat
==>
   =goal> ; modify
      judgment    yes)
```

Pattern 5	=g		?r			➔	=g				

The left side of the production specifies slot values in the goal buffer that will be matched. The retrieval buffer in the left side is queried by using "?", asking whether a chunk to retrieve is found or not. Then, the right side of the production changes the slot value in the goal buffer.

```
(P fail
   =goal>
      ISA         is-member
      object      =obj1
      category    =cat
      judgment    pending

    ?retrieval> ; no chunk could be found, retrieval has failed
      state       error
==>
   =goal>
      judgment    no; change the judgment slot to be no)
```

Pattern 6	=g	=v				➔	=g				+i

This pattern of the production reads information from the screen, and the information is placed into the imaginal buffer. In the left side of the production rule below, the goal is to read a letter and the visual buffer specifies a letter in a slot. Then, the right side of the production changes the goal to respond and put the visual information into the imaginal buffer.

```
(P encode-letter
   =goal>
      ISA         read-letters
      state       attend
   =visual>
      ISA         text ;a letter encoded from the screen
      value       =letter
==>
   =goal>
      state       respond
   +imaginal>
      isa         array
      letter      =letter)
```

Pattern 7	=g	=v		?i		➔	=g				+i

This production harvests the visual object that was placed into the visual buffer. It makes a request to the imaginal buffer to create a new chunk holding a representation of the letter.

```
(P encode-first-letter
    =goal>
       ISA         read-letters
       state       attend
    =visual>
       ISA         text
       value       =letter
    ?imaginal>
       buffer      empty
       state       free
==>
    =goal>
       state       start
    +imaginal>
       isa         array
       letter1     =letter)
```

Pattern 8	=g	=v		=i		➔	=g				=i

This production sets the `letter2` slot of the `array` chunk that is in the imaginal buffer to the letter that was read from the screen.

```
(P encode-second-letter
    =goal>
       isa        read-letters
       state      attend
    =visual>
       isa        text
       value      =letter
    =imaginal>
       isa        array
       letter2    nil
==>
    =goal>
       state      start
    =imaginal>
       letter2    =letter)
```

Pattern 9	=g					➔	=g	+vl			

If the goal is to read a letter, then the production is making a request to the visual-location buffer and changes the goal state to find out the location.

```
(P find-unattended-letter
   =goal>
      isa         read-letters
      state       start
 ==>
   +visual-location>
      ISA         visual-location
      :attended   nil
   =goal>
      state       find-location)
```

Pattern 10	=g	=vl ?v				➔	=g	+v			

If the goal is to read a letter, the location of the visual object is identified, and the visual buffer is ready to move attention, then this production moves attention to attend the visual objects.

```
(P attend-letter
   =goal>
      ISA         read-letters
      state       find-location
   =visual-location>
      ISA         visual-location
   ?visual>
      state        free
==>
   +visual>
      ISA         move-attention
      screen-pos  =visual-location
   =goal>
      state       attend)
```

Pattern 11	=g			=i	?m	➔	=g					+m

This pattern can be used when the production needs to make a motor action based on encoded visual information.

```
(P respond
   =goal>
      ISA         read-letters
      state       respond
   =imaginal>
      isa         array
      letter      =letter
   ?manual>
      state       free
==>
   =goal>
      state       done
   +manual>
      ISA         press-key
      key         =letter)
```

Pattern 12		?vl				➔		+vl			+i

This pattern can be used whenever the screen has recently changed, this pattern of the production is fired. In the condition part, using the buffer, like ?visual-location>, is a way to test whether a new display has been presented.

```
(P find-person
     ?visual-location>
       buffer      unrequested
   ==>
    +imaginal>
       ISA         comprehend-sentence
   +visual-location>
       ISA         visual-location
       > screen-x    105
       < screen-x    135)
```

Pattern 13		=vl ?vl ?v				➔		+v			

This pattern of the production makes a shift of visual attention after harvesting the requested visual location.

```
(P attend-visual-location
   =visual-location>
       ISA         visual-location
   ?visual-location>
       buffer      requested
   ?visual>
       state       free
   ==>
   +visual>
       ISA         move-attention
       screen-pos  =visual-location)
```

4.2 ACT-R's Subsymbolic Equations

This section summarizes equations used in the ACT-R subsymbolic construct that quantitatively controls many of the symbolic representations of the production system.

4.2.1 Activation and Base-level Learning Equations

The ability to retrieve a chunk is associated with the activation equation, also called memory strength function. The activation of a chunk i is represented as A_i. The activation equation consists of the base level activation and a noise component.

$$A_i = B_i + \varepsilon \qquad \textbf{Equation 4.1.}$$

The base level activation is represented as:

$$B_i = \beta + \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \qquad \textbf{Equation 4.2.}$$

β: a constant that is determined by the :blc parameter
n: the number of presentations for a chunk i
t_j: the time since the jth presentation
d: the decay parameter that is set using the :bll parameter

In ACT-R, the base-level activation is dependent on how often (frequency) and how recently (recency) a chunk is used. The base-level learning (BLL) supports two psychological laws of human memory: one is the Power Law of Learning and the other is the Power Law of Forgetting.

4.2.2 Recall Probability Equation

In ACT-R, a chunk will be retrieved if its activation is above the retrieval activation threshold, τ. The probability of a chunk's retrievability is represented by the expected activation, A.

$$P(A) = \frac{1}{1 + e^{\frac{\tau - A_i}{s}}} \qquad \textbf{Equation 4.3}$$

As the activation of a chunk, A_i, becomes higher, the recall probability approaches 1. The noise parameter, s, controls the sensitivity of recall in activation.

4.2.3 Retrieval Latency Equation

The activation determines a time how quickly a chunk can be retrieved. F is the latency factor parameter. A is the activation of the chunk to be retrieved.

$$T = Fe^{-A}$$ **Equation 4.4**

4.2.4 Spreading Activation Equation

The latency of a chunk in memory is determined by its level of activation (Anderson & Reder, 1999). Chunks in declarative memory spread activation to their associative links. The activation equation of a chunk i with the spreading activation is represented as:

$$A_i = B_i + \sum_k \sum_j W_{kj} S_{ji} + \varepsilon$$ **Equation 4.5**

W_{kj}: the amount of activation from source j in buffer k
S_{ji}: the strength of association between source j and chunk i.
B_i: the base-level activation
ε: the noise value

4.3 ACT-R's Mechanisms for Learning and Forgetting

The review of ACT-R's mechanisms is presented in this section. Based on the basic equations, ACT-R is able to learn and forget with some limitations.

4.3.1 Declarative Learning

The activation equation explains the learning of declarative chunks. Based on this equation, the base level learning and spreading activation are elaborated.

Base Level Learning with Activation

The base-level learning equation (Equation 4.2) and the activation equation (Equation 4.1) can describe a learning behavior representing whenever a chunk is presented, the base-level activation increases, and then decreases as a power function of the time.

Spreading Activation

I have briefly explained the fan effect in Chapter 2. In this section, let us look at computational mechanisms of the spreading activation. In the context of the fan effect, the basic activation equation (Equation 4.1.) can be expanded to incorporate the strength of association and the amount of attention given to a source. The activation, A_i, is as follows:

$$A_i = B_i + \sum_j W_j S_{ji} \qquad \textbf{Equation 4.6}$$

The base-level activation (B_i) of a chunk i reflects its past recency and frequency. The summation provides the sources of activation. In the fan effect experiment, the sources include people (e.g., a hippie, a doctor, etc), locations (e.g., park, church etc), and a preposition (e.g., in). W_j indicates the amount of activation from a source j, and S_{ji} indicates the association strength between a source j and a fact (or chunk) i.

In the ACT-R theory, the strength of association (S_{ji}) decreases as a logarithmic function of the fan as below:

$$S_{ji} = S - \ln(f_j) \qquad \textbf{Equation 4.7}$$

S is a constant and can be set with the maximum associative strength (:mas) parameter and f_j is the number of chunks where j is the value of a slot plus one for chunk j being associated with itself.

4.3.2 Declarative Forgetting

Pavlik and Anderson (2005) captured three major effects of memory in ACT-R. The effects are recency, frequency, and spacing effects in learning of Japanese-English vocabulary paired associates. The recency effect indicates performance is better the more recently an item in memory is practiced. The frequency effect indicates performance is better the more frequently an item in memory is practiced. The spacing effect is relatively related with practice and retention intervals.

Particularly, Pavlik and Anderson's modeling efforts deal with the different rate of forgetting depending on the spacing of practice over time. Basically, Pavlik and Anderson used an activation-based memory model, that is, each time a memory item is practiced, it receives an increment of strength but that these increments decay as a power function of time. In Equation 4.8, m_n indicates the memory strength of an item n as a function of times (t_is) after n prior presentations. The decay parameter of d is a constant value. This equation produces the power law of practice and forgetting. Figure 4.2 shows the activation curve.

$$m_n(t_{1...n}) = \ln\left[\sum_{i=1}^{n} t_i^{-d}\right] \qquad \textbf{Equation 4.8}$$

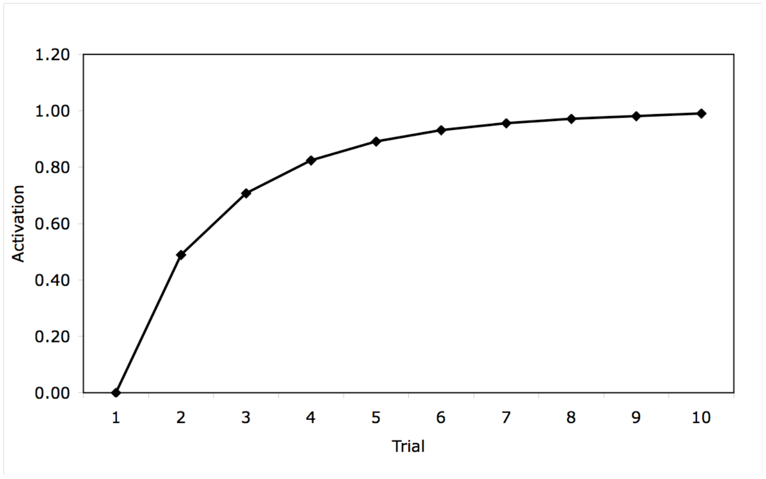


Figure 4.2. The activation curve, when the decay parameter is 0.2.

Pavlik and Anderson pointed out that this equation (Equation 4.8) of memory strength has three features:

(a) Define initial learning given prior practice

(b) Explain the evolution of forgetting over time with the constant decay parameter

(c) Explain integrating effects by summing each discrete practice

In terms of the activation value, m, as given by Equation 4.8, Pavlik and Anderson considered the probability of recall (P_r) as shown in Equation 4.9.

$$P_r(m) = \frac{1}{1 + e^{\tau - m/s}} \qquad \textbf{Equation 4.9}$$

In this equation, τ is the threshold parameter and s is the noise parameter. If the activation value increases, the recall probability approaches to 1, whereas, when the threshold value (τ) increases, the recall probability decreases, as shown in Figure 4.3. When the threshold value is equal to the activation value (i.e., $\tau = m$), the recall probability is 0.5.

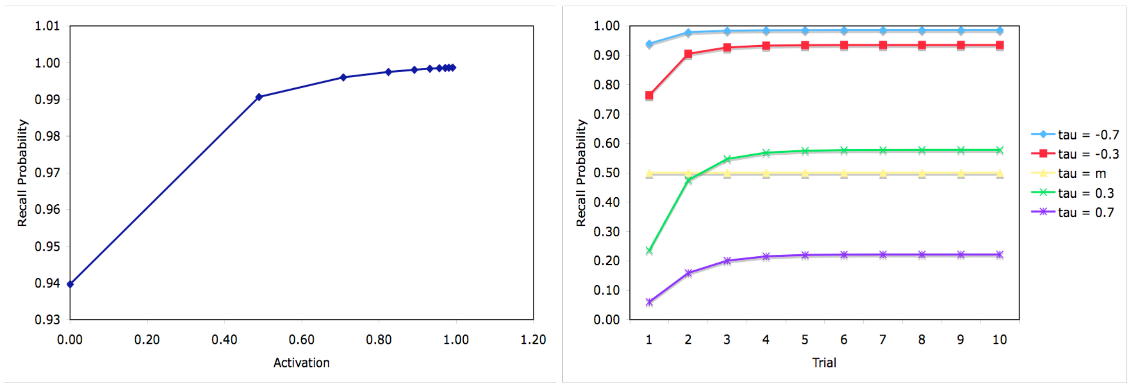


**Figure 4.3. Recall probability vs. activation,
and recall probability vs. trial numbers by varying values of τ.**

4.3.3 Procedural Learning

There are two types of learning mechanisms in ACT-R. The mechanisms including production compilation and utility learning cope with learning of production rules.

Production Compilation

Production compilation in ACT-R is a mechanism to learn new production rules by collapsing two productions into a single production. Basically, forming a new production is to combine the tests in the IF part of production rules to a single set of tests and to combine the actions in the THEN part of productions to a single set of actions. This production compilation mechanism can be classified into two types of buffers: (a) internal buffer and (b) external buffer as shown in Figure 4.4.

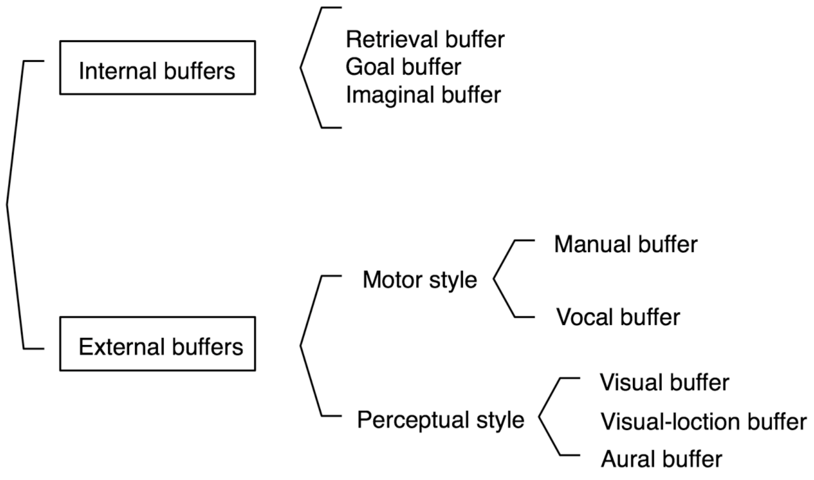
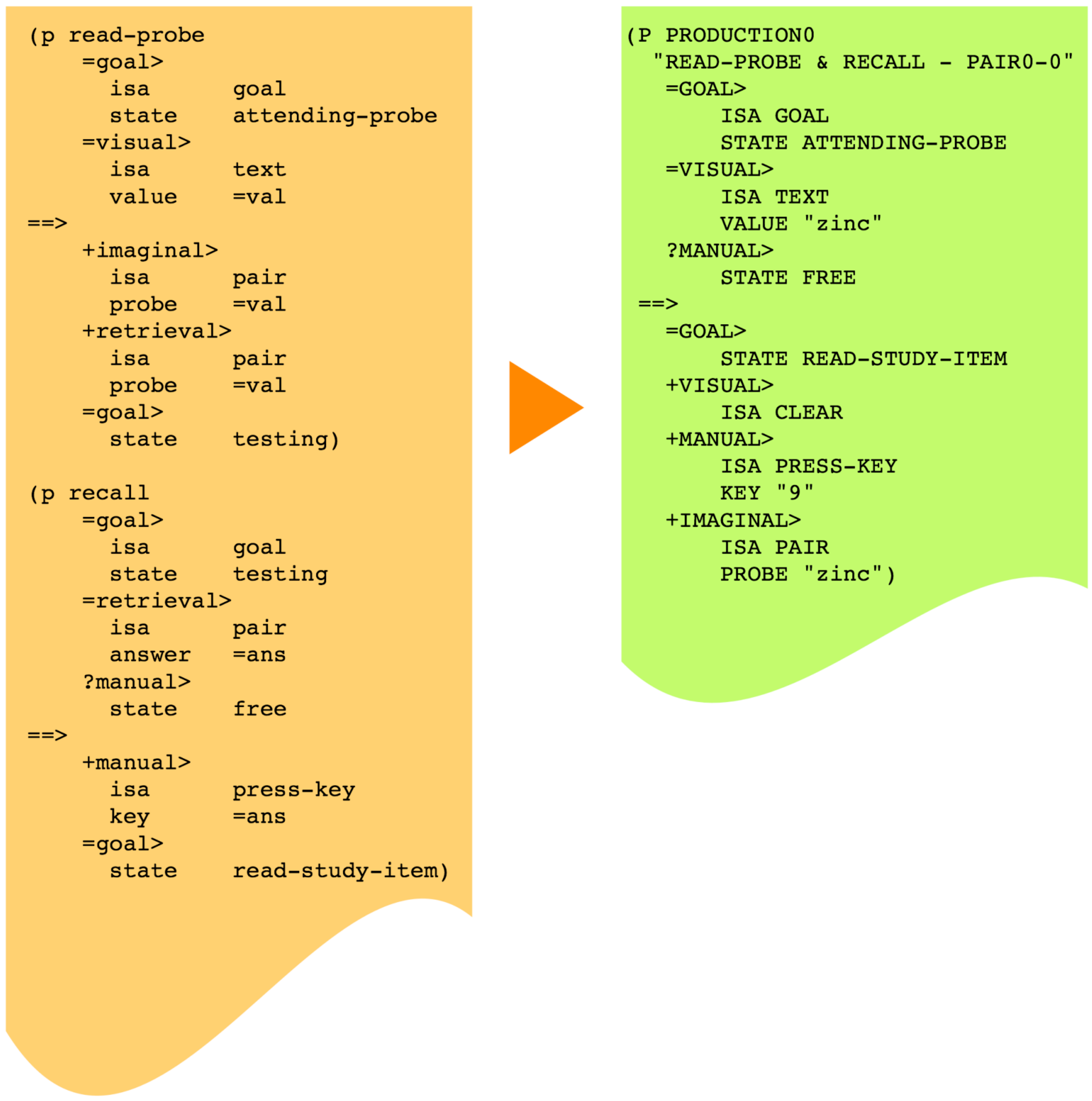


Figure 4.4. Internal and external buffer classification.

Internal buffers are not subject to change by the outside world, while external buffers are subject to change by the outside world. External buffers make requests for generating actions. Thus, if two productions make those actions at the same time, the ACT-R system produces jamming of those requests. Motor style buffers differ from the ones of perceptual style in that the motor style buffer will never hold a chunk but the perceptual style buffer will hold chunks that are based on information in the external world.

Production compilation in the retrieval buffer occurs when the first production's action is to request a retrieval and the second production's condition tests whether that retrieval is successful or not. These two productions are compiled by replacing variables in the retrieval request with constant values that are retrieved, forming a new production. However, if there is a retrieval error, compilation of productions are not allowed because it is not safe to predict a retrieval error. Here is an example of retrieval buffer style production compilation. The two productions are to fire successively and to finally retrieve a paired associate, as shown in Figure 4.5.

In Figure 4.5, the production compilation produces a single production by combining the read-probe and recall productions. In the newly compiled production, the retrieval request in the action part of the read-probe production is omitted but the imaginal request is not omitted.



```
(p read-probe
    =goal>
      isa       goal
      state     attending-probe
    =visual>
      isa       text
      value     =val
==>
    +imaginal>
      isa       pair
      probe     =val
    +retrieval>
      isa       pair
      probe     =val
    =goal>
      state     testing)

(p recall
    =goal>
      isa       goal
      state     testing
    =retrieval>
      isa       pair
      answer    =ans
    ?manual>
      state     free
==>
    +manual>
      isa       press-key
      key       =ans
    =goal>
      state     read-study-item)
```

```
(P PRODUCTION0
  "READ-PROBE & RECALL - PAIR0-0"
   =GOAL>
       ISA GOAL
       STATE ATTENDING-PROBE
   =VISUAL>
       ISA TEXT
       VALUE "zinc"
   ?MANUAL>
       STATE FREE
 ==>
   =GOAL>
       STATE READ-STUDY-ITEM
   +VISUAL>
       ISA CLEAR
   +MANUAL>
       ISA PRESS-KEY
       KEY "9"
   +IMAGINAL>
       ISA PAIR
       PROBE "zinc")
```

Figure 4.5. An example of production compilation of the retrieval buffer style[3].

Utility Learning

Production compilation creates a new production by collapsing old productions. To select the new production rather than an old one, ACT-R determines the selection by utilities. Like the activation mechanism, productions have their own utility values. Based on these utility values, one production can be preferred and be selected over another.

[3] This example is excerpted from the ACT-R tutorial, unit 7.

Also, the utilities can be learned from experience. If we let the expected utility as U_j, the probability of choosing a production i is:

$$\text{Probability}(i) = \frac{e^{U_i / \sqrt{2}s}}{\sum_j e^{U_j / \sqrt{2}s}} \qquad \textbf{Equation 4.10}$$

In the denominator of Equation 4.10, the summation indicates the sum of all productions that can currently be fired. That is, their conditions are satisfied during the match.

The utility values can be set by the modeler using :u parameter in the ACT-R model. The utilities of productions can be dynamically adjusted in terms of the reward they receive. This is called the utility learning. Let $U_i(n-1)$ is the utility of a production i after its $n-1^{\text{st}}$ application and $R_i(n)$ is the reward the production receives for its n$^{\text{th}}$ application. The utility, $U_i(n)$, after its n$^{\text{th}}$ is:

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \qquad \textbf{Equation 4.11}$$

In Equation 4.11, the value of α indicates the learning rate, which is typically set to 0.2 using :alpha parameter with sgp command in the ACT-R architecture.

4.3.4 Procedural Forgetting

As we reviewed previously, learning and forgetting in declarative memory can explained by the activation mechanism. Production rules are compiled to show speedup effects of practice. The question is how compiled production rules fall back to decreased task completion time due to a period of skill disuse.

I am exploring the ACT-R's learning mechanism. D. Bothell at Carnegie Mellon University (personal communication, April 4, 2008) mentioned that some negative or lower values of reward could lead to the compiled productions having a lower utility than the originals.

4.4 The Skill Retention Cognitive Model: Evolutionary Development Process

I describe the model development process in this section. I originally took an approach to develop a complete model in one step. However, it turned out that the model-building approach that I was taking was not successful very well. I scraped the previous approach and took a cyclic nature of the spiral development process. Boehm and Hansen (2001) stated that the spiral development process has been successfully used in various projects of military defense and commerce. Boehm and Hansen defined the spiral development model as:

> The spiral development model is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software-intensive systems. It has two main distinguishing features. One is cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

Basically, the spiral model expands a software system from its earlier versions of development process in an iterative manner. This process has been reinforced by the Department of Defense that included evolutionary acquisition process in its regulations, because the process of evolutionary acquisition with the spiral development can identify avoidable hazardous factors.

4.4.1 The Spiral Model Development Process

Risk factors can cause development process to be digressive from the system objective and its development process (Boehm & Hansen, 2001). The cognitive model that I am striving for developing has several risk factors resulting in from trivial to fatal failure. I frankly specify the entire risks factors here in Table 4.2.

Table 4.2 shows identified risk factors in developing a cognitive model of the skill retention. The envisioned goal in my dissertation is to model procedural skill retention but there is an enormous technology gap between the goal and the current status. It can possibly bridge the technology gap via the spiral process of development.

Table 4.2. Risk factors in the development process of the skill retention model.

Risk Factors	Likelihood	Impact	Mitigation Strategy
Modeling task knowledge and skills	Medium	Medium	Practice more for advanced techniques of using production systems
Representation of motor performance	Medium	Medium	Use the current capability with limitation but consider some extension of keystrokes (e.g., while holding a key down and pressing another down)
Representation of visual performance	Low	Low	N/A
Modeling and controlling long repeated task	High	High	Use meta-process of the ACT-R software that runs models asynchronously
Simulating timing characteristics of performance	High	High	Extend the ACT-R architecture
Modeling learning effects	Medium	High	Extend the ACT-R architecture
Modeling forgetting effects	High	High	Extend the ACT-R architecture

Figure 4.6 shows the spiral development process. Boehm and Hansen (2001) originally consider time and cost in two-dimensional axes. Based on this, I added the third axis of *technology gap* that we need to be aware of because identification of the technology gap helps developers to deeply comprehend concrete risk factors. In Figure 4.6 (b), each quadrant indicates a stepwise process of the spiral development. In Quadrant II, one can determine an objective of the system and consider alternatives and their constraints for that system. In Quadrant I, based on the risk analysis, one can provide a prototype in terms of the goal. Then, in Quadrant III, one can identify requirements and validation of those requirements while pursuing a goal to implement the system. In

Quadrant IV, one can plan for the next phase of the spiral development process by specifying goals, risks, requirements, validation, design validation, and implementation.

Based on the concept of the spiral model, I specify strategic processes to develop the skill retention model. For the first cyclic spiral stage, I implement an ACT-R model doing the first subtask of *OPEN FILE* out of 14 subtasks because modeling all of 14 subtasks would produce some risks, requiring much more production rules to create, and making the model be harder to debug and validate. Using the model of the first subtask, I explore the ACT-R's learning and forgetting mechanisms to test the ACT-R theory against human data. This first cyclic spiral stage will provide some useful insights and understandings to get ahead toward the ultimate goal of developing the skill retention model and implementing the final model with 14 subtasks.

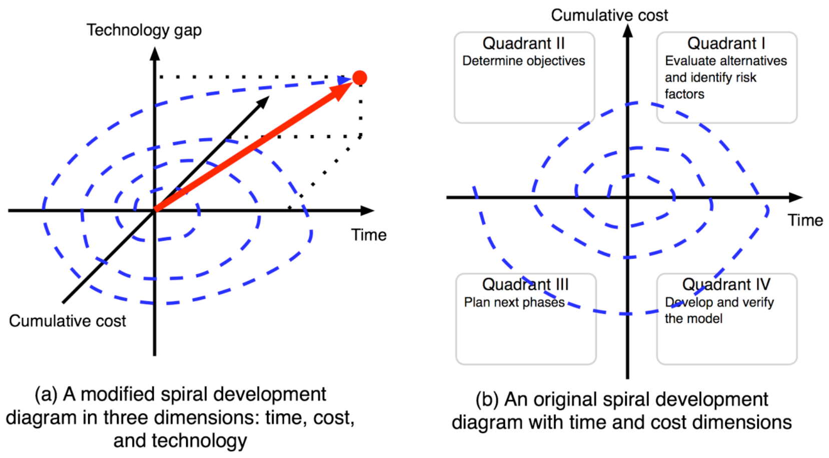

Technology gap
Time
Cumulative cost
(a) A modified spiral development diagram in three dimensions: time, cost, and technology
Cumulative cost
Quadrant II
Determine objectives
Quadrant I
Evaluate alternatives and identify risk factors
Time
Quadrant III
Plan next phases
Quadrant IV
Develop and verify the model
(b) An original spiral development diagram with time and cost dimensions

Figure 4.6. The spiral development process of the skill retention model.

4.4.2 Knowledge Representations in the Model

As I described in the section 2.7, there are 14 subtasks in the task. Each subtask has a subgoal. For example, the first subgoal is to open a Dismal spreadsheet file. The English rendition of this subgoal production would look like:

If the goal is open a Dismal spreadsheet file
Then move the mouse to the menu and click the appropriate file to open

Application of a production is viewed as a step of cognition (Anderson, 1982). Thus, I need to break down the fist subgoal into several cognitive contingencies that can represent the cognitive steps taken by users. I tried to decompose a subgoal into smaller steps of cognition. For the subgoal of opening a file, humans would do like this: think about how to open a file, attend to a menu item, move attention, and make some corresponding actions (e.g., mouse click, mouse move, or keypress). Table 4.3 provides English-like descriptions of productions in the left column and corresponding productions rules of ACT-R in the right column.

Table 4.3. Example productions for the Dismal spreadsheet task.

English-like Descriptions	ACT-R Productions

If the goal is to do the subtask1
the step slot is getting-ready
and the chunk in the retrieval buffer is of type operator
the pre slot has a value called =state
the action slot has get-ready
the post slot has a value of =post
 	
then,
 	
request a retrieval of a chunk in the operator chunk type to find the value of =post in the pre slot,
and change the value of the step slot in the goal buffer
	`(P getting-ready`
	`=goal>`
	`isa task`
	`step getting-ready`
	`=retrieval>`
	`isa operator`
	`pre =state`
	`action get-ready`
	`post =post`
	`;=visual-location>`
	`isa visual-location`
	`;?visual> state free`
	`==>`
	`;+visual>`
	`isa move-attention`
	`screen-pos =visual-location`
	`+retrieval>`
	`isa operator`
	`pre =post`
	`=goal>`
	`step attending)`

If the goal is to do the subtask1
the step slot is attending
and the chunk in the retrieval buffer is type operator
the pre slot has =post
the action slot has attend
the post slot has move-to-file
there is a chunk type of visual-location in the visual location buffer,
check the visual performance is available
then,
request attention movement to =visual-location in the visual buffer
request a retrieval of a chunk type operator to find the slot value of move-to-file
and, change the value of the step slot in the goal buffer
	`(P attend-to-file`
	`=goal>`
	`isa task`
	`step attending`
	`=retrieval>`
	`isa operator`
	`pre =post`
	`action attend`
	`post move-to-file`
	`=visual-location>`
	`isa visual-location`
	`?visual>`
	`state free`
	`==>`
	`+visual>`
	`isa move-attention`
	`screen-pos =visual-location`
	`+retrieval>`
	`isa operator`
	`pre move-to-file`
	`=goal>`
	`step moving)`

4.4.3 Visual Perception and Motor Performance in the Model

This section describes visual perception and motor performance of the model from the perspective of the ACT-R architecture. Gray and Altman (2001) provided a useful framework of understanding interactive behavior by illustrating a triad, that is, *Cognition*, *Artifact*, and *Task*. There are several reasons why we need to consider the triad. One might conduct experiments by using simple tasks and only focus on cognition and task, disregarding the role of artifact. Also, one might only focus on the artifact, disregarding interaction with users. Also, one might focus on development of the artifacts, in response to tasks, but generally not in response to cognitive concerns. Gray and Altman (2001) argued that the price of ignoring any one of cognition, artifact, and task cause interactive human behavior to be more taxing with effort and be more error-prone.

Byrne (2001) also reinforced that interactive behavior of a user interacting with an interface is a function of the properties of three things: the cognitive, perceptual, and motor capabilities of the user, termed *Embodied Cognition,* the *Task* the user is engaged in, and the *Artifact* the user is employing to do the task. Gray and Altman simply referred to *Embodied Cognition* as *Cognition*, indicating a broad meaning of cognition, that is, indicating not only cognitive capabilities and limitations of human but also the perceptual-motor capabilities. In this dissertation, I prefer to use embodied cognition to indicate cognitive and perceptual-motor capability of a cognitive model.

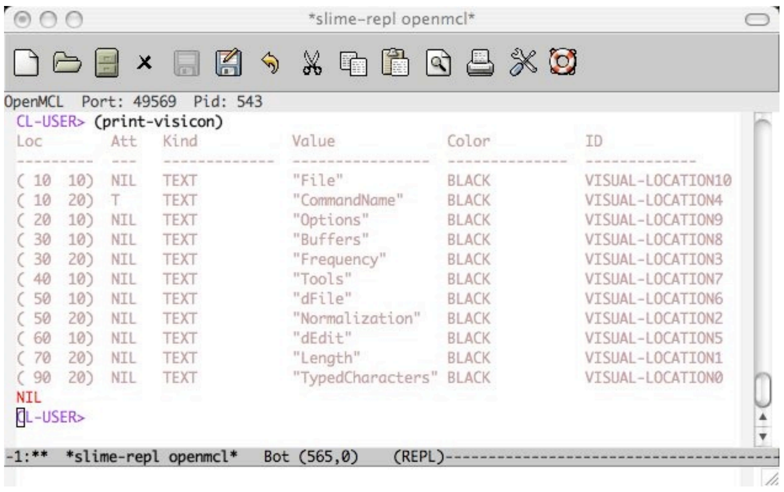
Back in 1998, there has been a successful effort to extend the ACT-R architecture to describe and predict human behavior in primarily cognitive domains with a perceptual/motor system (Byrne, 2001; Byrne & Anderson, 1998). ACT-R provides the capability of the embodied cognition that is mostly affected by the Kieras and Meyer (1997) EPIC architecture. As shown in Figure 2.4, the production system in the procedural module is the central to four perceptual-motor modules such as, visual, manual, vocal, aural modules. In this dissertation, I only consider visual and manual modules in ACT-R because human subjects performed given tasks without using any vocal and aural activities.

Basically, all modules can be queried to find out each module's current internal state, such as *state free* or *state busy*. Furthermore, the perceptual motor modules have more complicated internal systems including preparation, processor, and execution. These internal systems can be individually queried where their states are *free* or *busy*.

Visual Perception Performance in the Model

The visual module equips the model to see objects in the external world that the model interacts with. Particularly, the visual module consists of two subsystems of "where" and "what".

Visual objects in the external world can have one or more features, such as location, size, or color. The features are stored in a place called the *visual icon*. Based on these features, the visual module creates chunks, providing declarative memory representations of the visual scene. Then, those chunks can be matched by production rules. A production rule can specify location, size, or color of the visual object in its IF condition part to constrain the acceptability of rule match. After the chunk matching with a production rule, that chunk became completely developed and is added to declarative memory. Here is example visual information that the model sees in the external world of the Dismal spreadsheet window in Figure 4.7.



```
OpenMCL  Port: 49569  Pid: 543
 CL-USER> (print-visicon)
 Loc        Att  Kind          Value            Color          ID
 ---------  ---  ------------  ---------------  -------------  -------------
 ( 10  10)  NIL  TEXT          "File"           BLACK          VISUAL-LOCATION10
 ( 10  20)  T    TEXT          "CommandName"    BLACK          VISUAL-LOCATION4
 ( 20  10)  NIL  TEXT          "Options"        BLACK          VISUAL-LOCATION9
 ( 30  10)  NIL  TEXT          "Buffers"        BLACK          VISUAL-LOCATION8
 ( 30  20)  NIL  TEXT          "Frequency"      BLACK          VISUAL-LOCATION3
 ( 40  10)  NIL  TEXT          "Tools"          BLACK          VISUAL-LOCATION7
 ( 50  10)  NIL  TEXT          "dFile"          BLACK          VISUAL-LOCATION6
 ( 50  20)  NIL  TEXT          "Normalization"  BLACK          VISUAL-LOCATION2
 ( 60  10)  NIL  TEXT          "dEdit"          BLACK          VISUAL-LOCATION5
 ( 70  20)  NIL  TEXT          "Length"         BLACK          VISUAL-LOCATION1
 ( 90  20)  NIL  TEXT          "TypedCharacters" BLACK         VISUAL-LOCATION0
 NIL
 CL-USER>
-1:**  *slime-repl openmcl*   Bot (565,0)    (REPL)------------------------------------
```

Figure 4.7. Visual objects in the real task environment that the model sees.

As shown in Figure 4.7, a visual object's location is represented by a x-y coordinate in pixels. Shifting attention from one location to another occurs asynchronously with respect to the production system (Byrne, 2001) and takes 185 ms based on previous theoretical research (see Anderson, Matessa, & Lebiere, 1997). Thus, all information in the Dismal buffer in Emacs window should be converted to a pixel location and be delivered to the model. ESEGMAN is to take the role of transfer and delivery of the visual information to the model. More details are provided in Section 5.5.

Motor Performance in the Model

Modeling human performance requires a computational model to perform a given task like a human does the same task. In this section, I describe how the ACT-R model supports plausible and reasonable human motor performance.

The ACT-R motor module can only prepare one movement at a time. The manual buffer interacts with the motor module. The manual buffer never holds a chunk. It issues commands to query the motor module state.

The Dismal spreadsheet task contains multiple actions of key presses, mouse clicks, and mouse movements by users. In an ACT-R model, it is possible to specify where the model's hand should be located (i.e. at mouse or at keyboard). For the ACT-R model of mouse users, the model's hand is located at a mouse by using a command, (start-hand-at-mouse). When the model needs to generates key presses, it is necessary for the model to issue a command of moving the model's right hand to the home position, (7 4), on the virtual keyboard. Conversely, for the ACT-R model of keyboard users, it is not necessary to request the command of the model's hand at mouse. However, it is necessary to request a command that the model's right hand moves to the virtual mouse location, (28 2), before requesting mouse actions.

The Dismal task environment offers fixed-order menus. That is, the position of a target menu is known. In this case, the critical latency component can be attributable to Fitts' Law (Fitts, 1954) description of the motion (see Anderson & Lebiere, 1998). The execution time of motor movement based on Fitts' Law can be predicted as a function of the distance to the target (amplitude) and the width of the target (tolerance). ACT-R predicts movement time using Equation 4.12. The parameter of *b* coefficient (:MOUSE-FITTS-COEFF) in Fitts' equation is used when the model moves the mouse cursor for aimed movements. Its default value is 0.1 and can be set to any positive value.

$$T = b * \log_2(D/W + 0.5) \qquad \textbf{Equation 4.12}$$

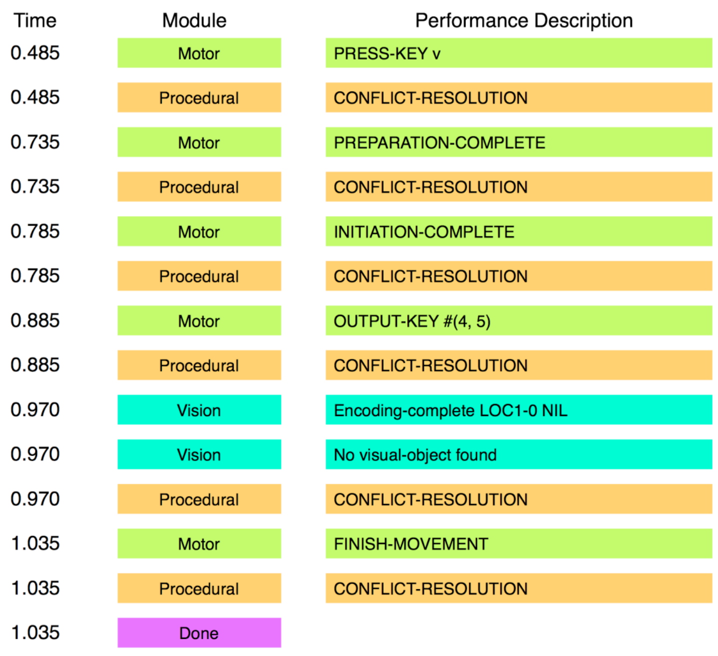
T = the time of the movement (seconds)

b = a parameter dependent on the type of motor action, called index of difficulty in sec/bit

D = the distance to the target

W = the width of the target

Figure 4.8 shows a list of trace of the ACT-R performance to press a key. The ACT-R model presses a v-lettered key that is located (4 5) at the ACT-R virtual keyboard (see Figure 4.2). This event took place at 0.485 s by firing a production, requesting a press-key. Then, it takes 250 ms to complete preparation of press-key at 0.735 s and 50 ms to initiate the action (initiation-complete) at 0.785 s, another 100 ms to strike the key (OUTPUT-KEY # (4 5)) at 0.885 s, and finally 150 ms for one finger to return to the home row (finish-movement) at 1.035 s.



Time	Module	Performance Description
0.485	Motor	PRESS-KEY v
0.485	Procedural	CONFLICT-RESOLUTION
0.735	Motor	PREPARATION-COMPLETE
0.735	Procedural	CONFLICT-RESOLUTION
0.785	Motor	INITIATION-COMPLETE
0.785	Procedural	CONFLICT-RESOLUTION
0.885	Motor	OUTPUT-KEY #(4, 5)
0.885	Procedural	CONFLICT-RESOLUTION
0.970	Vision	Encoding-complete LOC1-0 NIL
0.970	Vision	No visual-object found
0.970	Procedural	CONFLICT-RESOLUTION
1.035	Motor	FINISH-MOVEMENT
1.035	Procedural	CONFLICT-RESOLUTION
1.035	Done	

Figure 4.8. An example of ACT-R's motor performance pressing a key.

In the Dismal task, users press a key and hold that key while pressing the other key. For example, a user presses C-x (press and hold the control key and press x). For this task, that I call an Emacs-type key press, the timing characteristics are not supported by the current ACT-R architecture. As a future work, it is necessary to extend the ACT-R's motor module performance to include such Emacs-type "control" key presses. In this dissertation study, I will double the time that ACT-R uses to represent those key presses for a short-term plan, but, I will plan to gather data of Emacs-type key press to extend the ACT-R's motor performance for a long-term plan.

4.4.4 Can ACT-R Constructs Support a Loop of Repeated Tasks?

The Dismal spreadsheet task is procedural skill requiring a set of sequential procedures. Also, the spreadsheet task requires a user to perform a repeated task. For example, the subtask3, that is calculating *frequency*, requires a user to repeat the calculation five times. To achieve these procedural and repetitive characteristics of the task, I used a special chunk type and chunks in the model.

In the model, I created a chunk type named *operator*. Each operator tells the model what to do in various states while the model is performing the given task. The operator chunk type consists of 6 slots and the model is comprised of 11 operators. The *pre* slot indicates what state the model is in and the *post* slot indicates what state will occur after the action. The *action* slot indicates what action to be taken. The *object1*, *object2*, and *object3* slots indicate possible objects that are needed for the model's performance. Table 4.4 shows the operator chunk type and chunks.

Table 4.4. The model's chunk-type and chunks that guide the flow of the sequential task.

```
Chunk-type
(chunk-type operator pre action object1 object2 object3 post)

chunks for operator chunk-type
(op1-sub1 isa operator pre start action get-ready post attend-to-file)

(op2-sub1 isa operator pre attend-to-file action attend object1 file
post move-to-file)

(op3-sub1 isa operator pre move-to-file action move object1 file post
click-on-file)

(op4-sub1 isa operator pre click-on-file action click object1 file post
attend-to-openFile)

(op5-sub1 isa operator pre attend-to-openFile action attend object1
openFile post move-to-openFile)

(op6-sub1 isa operator pre move-to-openFile action move object1
openFile post click-on-openfile)

(op7-sub1 isa operator pre click-on-openfile action click object1
openFile post attend-to-dismalFile)

(op8-sub1 isa operator pre attend-to-dismalFile action attend object1
normalization-dis post move-to-dismalFile)

(op9-sub1 isa operator pre move-to-dismalFile action move object1
normalization-dis post click-on-dismalFile)

(op10-sub1 isa operator pre click-on-dismalFile action click object1
dismalFile post click-to-choose)

(op11-sub1 isa operator pre click-to-choose action click post retrieve-
subgoal-2)
```

In addition to the chunk type of *operator* and chunks, I specified a slot named *step* in the goal buffer representing various subordinate states of each procedure. This *step* slot maintains information about what the model is doing, indicating explicitly which productions are appropriate each time. Here are example production rules.

```
(P start-task
     =goal>
     isa task
        subtask-previous nil
        subtask-current sub1
        state =state
        step ready
  ==>
     +retrieval>
      isa operator
        pre =state
     =goal>
        step getting-ready)

(P getting-ready
     =goal>
      isa task
        step getting-ready
     =retrieval>
      isa operator
        pre =state
        action get-ready
        post =post
  ==>
     +retrieval>
      isa operator
        pre =post
     =goal>
        step attending)
```

Similarly, it is possible to represent a loop of repeated task by the use of the *step* slot in the goal buffer. The *step* slot can indicate that which production rule is appropriate for the repetitive task cycle.

4.4.5 Learning and Forgetting in the Model

The first subtask of “Open file” was modeled within the ACT-R architecture. This model was attempted to represent a mouse user’s behavior. The model has 11 production rules to perform the given task. Each production was written to represent a cognitive unit of human behavior.

Learning

I attempted to represent learning of the Dismal spreadsheet task by running the model multiple times based on the ACT-R’s activation mechanism and the production compilation mechanism. The model is able to run multiple times without resetting the model’s states and conditions. That is, the current trial is affected by the previous state of the model to produce learning effects by each run. Each simulation trial indicates a training and test session of the task. However, when we interpret the model’s performance, it is necessary to consider whether a simulated trial can be a supplant for a real training session, like the description of psychological time vs. real time in simulation in the next section 3.5.6.

Forgetting

My goal is to have decreased performance of task completion time after some retention interval that is a period of skill disuse. Currently, ACT-R does not completely support this slowdown performance. The production compilation mechanism can create a new production rule representing speedup. However, the learned rules are not reversed to the original or even worse performance (i.e., increased task completion time).

To resolve this question, I had several discussions through emails with the ACT-R implementer, Dan Bothell, at Carnegie Mellon University (personnel communications, April 4, 2008). Based on his suggestions, I will explore skill retention and forgetting in a couple of ways. The time prediction must meet psychological plausibility and sensitivity of decay performance in terms of task types.

- **Using lower reward value**
  It is possible set a reward value to each production. Lower reward value could lead to the compiled productions having a lower utility value than the originals. This may be able to slow the task completion time.

- **Using a command, (run-full-time time) based on the activation mechanism**
  If a slowdown performance comes after a delay, then it might come out naturally from the declarative retrieval times. The activation of the chunks needed for the task would decay during the delay and thus the time would be longer until they've been used enough to boost their activation back up. To put a delay, I can simply add this command, (run-full-time 20)[4], to the model.

[4] This command will run the current process until either 20 s passes or a break event is executed.

4.4.6 Psychological Time vs. Real Time in Simulations

Anderson, Fincham, and Douglass (1999) investigated the strength of a memory trace in terms of various practices at time t_j—that is, strength is equal to $\sum t_j^{-d}$, representing the power law of practice and the power law of retention. However, Anderson et al. (1999) acknowledged that this equation could not fit retention data over long intervals of up to 14 months.

This problem was also pointed out by Pavlik (2005), and Pavlik and Anderson (2005). This problem necessitated an assumption, indicating that memory decay in a model would occur more slowly than one in the experimental sessions.

It is possible to slow down decay by using a scaling factor that is multiplied to the passage of time outside the experiment. This scaling of time is called "psychological time". In an ACT-R simulation, forgetting also depends on the psychological time between presentations rather than the real time (Pavlik Jr. & Anderson, 2005). They introduced a scaling parameter, h, in using the recall equation. The parameter, h, is multiplied to the time that occurs between sessions in the recall equation. An example of use is found in Pavlik and Anderson (2005).

4.5 The Model Prediction

The subtask of "Open file" was modeled in the standard ACT-R 6 architecture. With the ACT-R model that simulates a mouse user in the Dismal spreadsheet task, I explored the ACT-R theory to explain practice and skill retention for given intervals of knowledge and skill disuse.

4.5.1 Learning in the Model

I gathered a set of learning data by running one model subject for four trials in a succession. Particularly, ACT-R's subsymbolic computations were enabled (i.e. the production compilation and the activation mechanism enabled), when the model has run.

Parameters in the ACT-R modules were configured to predict learning performance. First of all, I enabled the subsymbolic computation by turning on the parameter, (:esc t). For the declarative module, the activation trace parameter (:act) was set to *t* and other parameters used the ACT-R's default settings, such as the activation noise parameter (:ans 0.15), the retrieval threshold parameter (:rt -0.5), the latency factor parameter (:lf 0.63), the base level learning parameter[5] (:bll 0.5), and the maximum associative strength parameter (:mas 1.6). For the production compilation module, the production learning parameter (:epl) and its trace (:pct) were enabled. For the utility module, all parameters were set to default values, such as the utility threshold parameter (:ut nil), the initial production utility value (:iu 10), the default action time, that is the amount of time between selecting and firing a production (:dat 0.05), the expected gain parameter (:egs 0.0), the utility learning flag parameter (:ul t), and the production learning rate parameter (:alpha 0.2).

The gathered data are reported in Table 4.5 and the plot for the data is seen in Figure 4.9. As you see in Figure 4.9, the task completion time increased at the second trial, but showed decrease in task time after the second trial.

Table 4.5. The model performance with practice for four serial trials (*n* = 1).

Trial	Time (s)
1	4.925
2	6.108
3	4.977
4	4.214

[5] The decay parameter (*d*) in the equation, $B_i = \ln(\sum_{j=1}^{n} t_j^{-d}) + \beta_i$, is set using the base-level learning parameter.

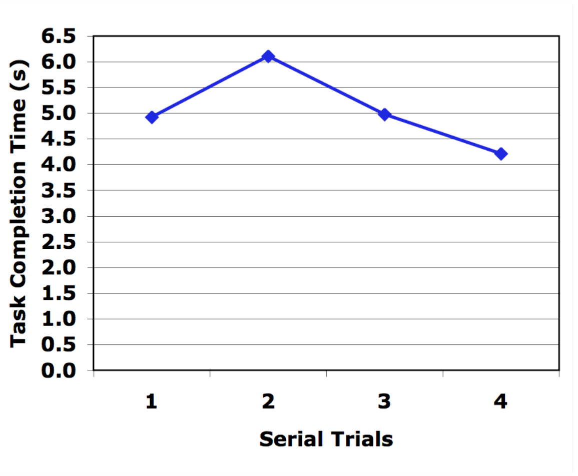
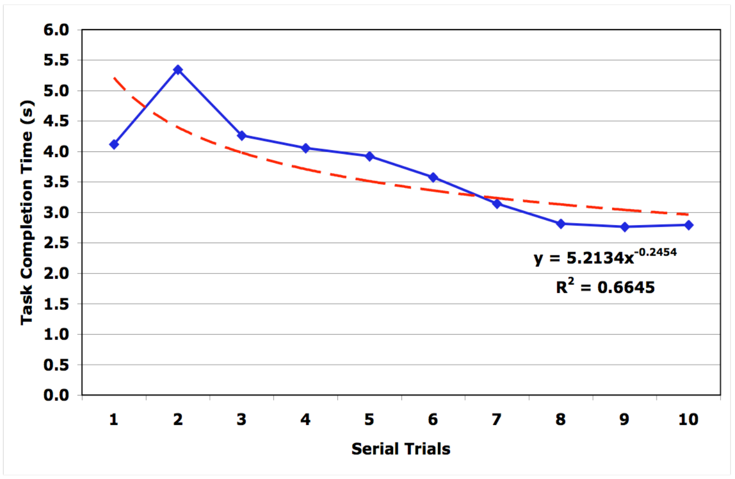
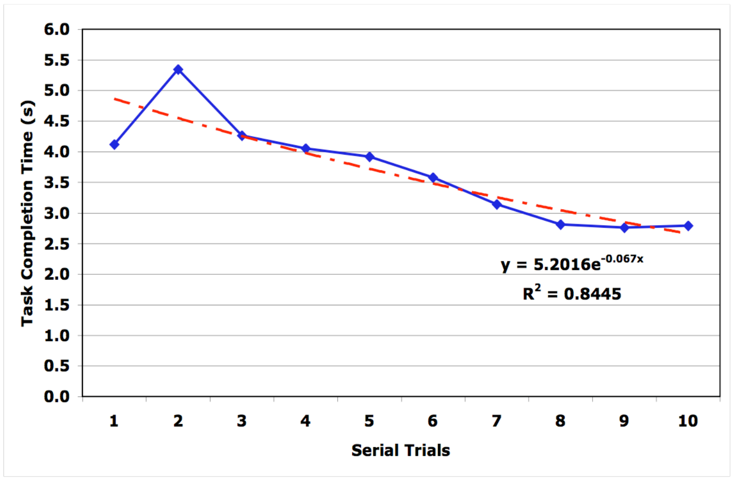


Figure 4.9. The model prediction of the task completion time by practice for four serial trials (*n* = 1).

I increased the number of serial trials to see more learning effects for more extended time duration. The model ran 10 serial trials. Figure 4.10 shows the performance of one model subject.



(a) Fitted with a power function.



(b) Fitted with an exponential function

Figure 4.10. A plot of task completion time with practice for ten serial trials and a fitted curve, a power curve and an exponential curve, (*n* = 1).

As the next step, I explored the model performance of multiple runs (i.e. four serial trials or ten serial trials) with ten model subjects. Also, I compared the model performance while both turning on production compilation by setting the parameter of enable production learning[6] and production compilation[7] to *t* and turning off production compilation by setting them to *nil*.

Table 4.7 shows descriptive data of the model's performance and Figure 4.11 shows their plots for ten model subjects during four serial trials. Ten model subjects have completed the task for four serial trials.

It was expected that the task completion time would decrease as a power function. However, regardless of enabling and disabling the production compilation, the task time increased more on the second trial that the one on the first trial in both cases. After the second trial, the task completion time decreased on the third and fourth trials. In Figure 4.11, you can see an increase in task completion time on the second trial. The observation of the model performance on the second trial caused the need to further investigate the learning mechanism and the model performance.

---

[6] *:epl*
[7] *:pct*

Table 4.7. Performance of ACT-R model subjects for four serial trials with production compilation on or off.

Production compilation – Off, 11 production rules before combining rules				
	Serial Trials (*n* = 10)			
	1	2	3	4
Mean	4.83	5.43	4.49	4.13
SD	0.50	0.66	0.48	0.54
Production compilation – On, 8 production rules after combining rules				
	Serial Trials (*n* = 10)			
	1	2	3	4
Mean	4.37	4.99	4.62	3.98
SD	0.35	0.36	0.35	0.29

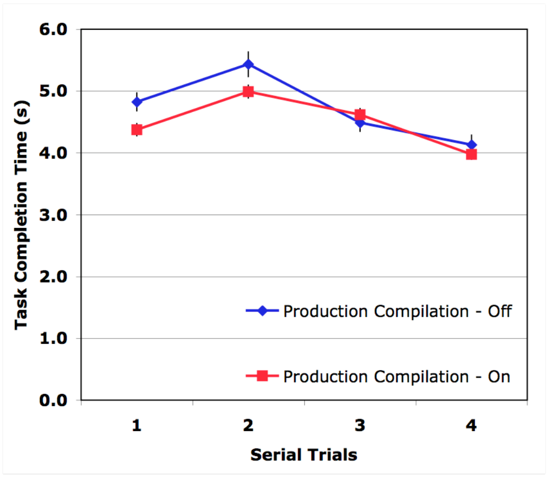


Figure 4.11. The mean task completion time over four serial trials with production compilation on (*n* = 10) and off (*n* = 10).

By enabling and disabling the production compilation mechanism, the number of production rules would be different. The model is comprised of 11 production rules in procedural memory and chunks in declarative memory. That is, the number of productions is 11 when the production compilation mechanism is disabled. The compilation process combines parent productions into a newly created production. From 11 production rules, the production compilation process produces 5 combined productions, as shown in Figure 4.12.

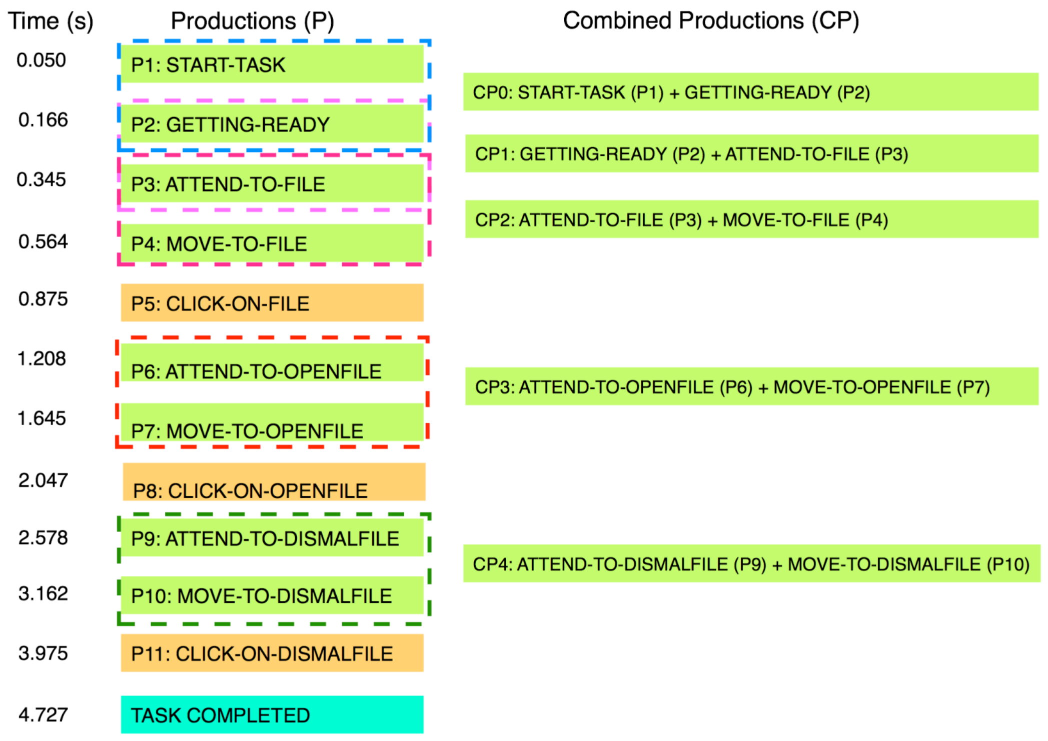
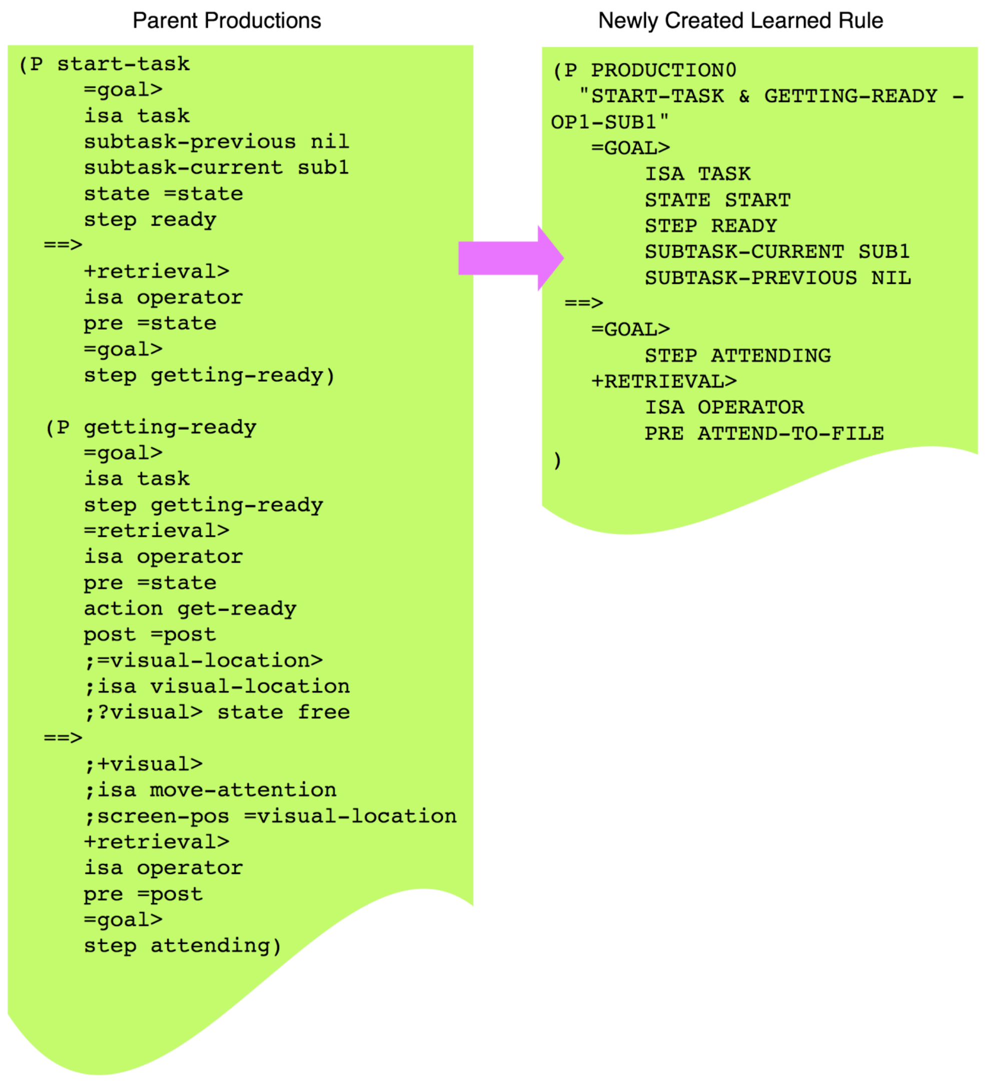


Figure 4.12. ACT-R model's production rules and combined rules by the production compilation mechanism.

For example, there are two separate productions of START-TASK and GETTING-READY. The ACT-R's production compilation mechanism combined these two productions into one production. Figure 4.13 shows the example production rules and a combined rule by ACT-R's learning mechanism.



```
(P start-task
     =goal>
     isa task
     subtask-previous nil
     subtask-current sub1
     state =state
     step ready
  ==>
     +retrieval>
     isa operator
     pre =state
     =goal>
     step getting-ready)

  (P getting-ready
     =goal>
     isa task
     step getting-ready
     =retrieval>
     isa operator
     pre =state
     action get-ready
     post =post
     ;=visual-location>
     ;isa visual-location
     ;?visual> state free
  ==>
     ;+visual>
     ;isa move-attention
     ;screen-pos =visual-location
     +retrieval>
     isa operator
     pre =post
     =goal>
     step attending)
```

```
(P PRODUCTION0
  "START-TASK & GETTING-READY -
OP1-SUB1"
   =GOAL>
       ISA TASK
       STATE START
       STEP READY
       SUBTASK-CURRENT SUB1
       SUBTASK-PREVIOUS NIL
 ==>
   =GOAL>
       STEP ATTENDING
   +RETRIEVAL>
       ISA OPERATOR
       PRE ATTEND-TO-FILE
)
```

Figure 4.13. An example of combined rules in the Dismal spreadsheet task by the production compilation mechanism. Several lines starting with ";" indicate commented-out codes that are to be added with ESEGMAN.

Table 4.8 summarizes gathered data from 10 model subjects for 10 serial trials. As shown in Figure 4.14, it produces very similar predictive performance whether the model was enabled or disabled by the production compilation. Interestingly, on the second trial, the mean task completion time went up and then decreased as time for serial trials has passed. This pattern is similar to the pattern with four serial trials shown in Figure 4.11.

Table 4.8. Performance of ACT-R model subjects for ten serial trials with production compilation on or off.

Production Compilation – Off										
	Serial Trials (n = 10)									
	1	2	3	4	5	6	7	8	9	10
Mean	4.63	5.25	4.84	4.24	3.73	3.39	3.32	3.06	2.95	2.85
SD	0.66	0.62	0.46	0.45	0.36	0.37	0.45	0.34	0.49	0.41
Production Compilation – On										
	Serial Trials (n = 10)									
	1	2	3	4	5	6	7	8	9	10
Mean	4.18	5.19	4.59	3.92	3.69	3.50	3.20	2.88	2.79	2.72
SD	0.35	0.39	0.33	0.30	0.28	0.29	0.16	0.22	0.22	0.22

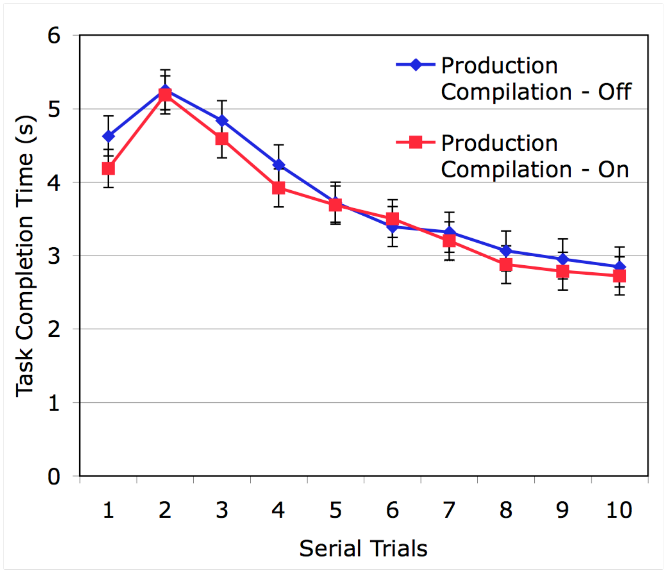


Figure 4.14. Mean task completion time over four serial trials with production compilation on (n = 10) and off (n = 10).

Table 4.9 shows the utility value in the production rule learning. Five combined productions have the same current utility value for each production (CP0 to CP4). Each time combined rules are recreated, the strength of the rules increases in terms of the equation, $U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)]$.

Table 4.9. Utility values during 10 serial trials with 10 subjects.

	Utility value									
	:utility					:u				
Trial	CP0	CP1	CP2	CP3	CP4	CP0	CP1	CP2	CP3	CP4
1	nil	nil	nil	nil	nil	0.000	0.000	0.000	0.000	0.000
2	-0.169	0.004	-0.099	0.144	0.184	2.000	2.000	2.000	2.000	2.000
3	2.175	2.015	2.017	1.914	1.780	3.600	3.600	3.600	3.600	3.600
4	3.578	3.774	3.579	4.045	3.516	4.880	4.880	4.880	4.880	4.880
5	4.830	4.858	4.693	4.313	5.208	5.904	5.904	5.904	5.904	5.904
6	5.538	5.893	5.870	5.868	5.590	6.723	6.723	6.723	6.723	6.723
7	6.435	6.588	6.312	6.763	6.734	7.379	7.379	7.379	7.379	7.379
8	7.339	7.522	7.269	7.311	7.408	7.903	7.903	7.903	7.903	7.903
9	7.911	7.923	8.045	7.685	8.097	8.322	8.322	8.322	8.322	8.322
10	8.156	8.246	8.299	8.529	8.390	8.658	8.658	8.658	8.658	8.658

Note: The parameter of :utility indicates the last computed utility value of the production during conflict resolution. The parameter of :u indicates the current $U(n)$ value for the production.

4.5.2 Learning in the Model Adjusted

In the previous section, unexpected learning performance was observed, that is, the task completion time went up on the second trial. To fix this problem, I adjusted parameter values of the Skill Retention Model, such as the decay parameter, the latency factor parameter, and the retrieval threshold parameter. The decay parameter was adjusted from 0.5 to 0.3, the latency factor parameter was adjusted from 0.63 to 1.9, and the retrieval threshold parameter value was adjusted from -0.5 to -1.0, as shown in Table 4.10. With the adjusted parameters, I collected fifteen model subjects' data, shown in Table 4.11. The plots are shown in

Table 4.10. Adjusted parameter values in the Skill Retention Model.

Parameter		Before	After
Decay parameter	:bll	0.5	0.3
Latency factor	:lf	0.63	1.9
Retrieval threshold	:rt	-0.5	-1.0

Table 4.11. Learning performance of the Skill Retention Model with adjusted parameter values

	Serial Trials (n = 15)			
	1	2	3	4
Mean	27.08	23.14	17.75	14.04
SD	2.31	1.87	1.20	1.05
SE	0.60	0.48	0.31	0.27

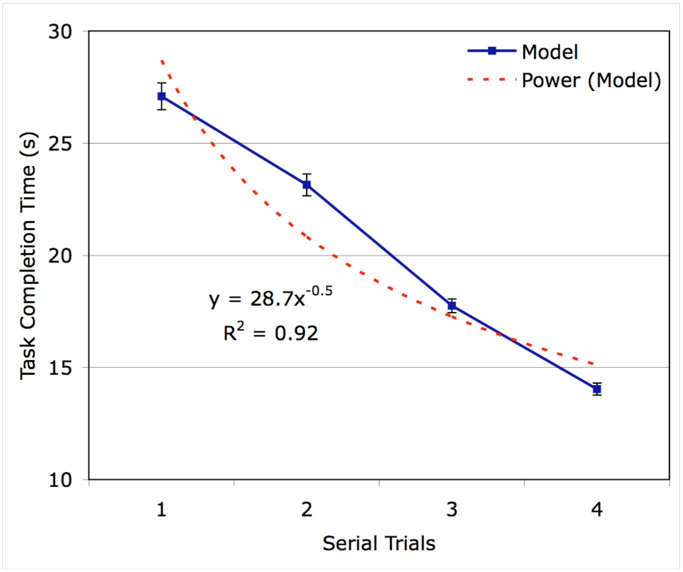


Figure 4.15. Learning performance of the Skill Retention Model based on adjusted parameters with standard error bars (*SE*).

4.5.3 Forgetting in the Model

I first considered setting a lower reward value to each production. A lower reward value could lead to the learned productions having a lower utility value than the originals. This may be able to slow the time to select a production. However, it is questionable that setting a reward value by a modeler is too subjective to represent a general performance of skill decay. I need to have founding guidelines to assigning a reward value to each production rule.

Thus, I tried to use the natural decay of the ACT-R model, using a command, (run-full-time T) based on the activation mechanism. This approach is basically slowing down the activation value of elements in declarative memory. If a slowdown performance comes after a delay, then skill decay might come out naturally from the declarative retrieval times. The activation of the chunks needed for the task would decay during the delay and thus the time would be longer until they've been used enough to boost their activation back up. To put a delay, I can simply add this command, (run-full-time T) to the model that ages the model by T seconds.

4.5.4 Issues Raised by the Model

I encountered several pivotal challenges in the modeling of skill retention. The ACT-R model produced predicted time to complete the given task, open a file. With respect to learning prediction, the model was expected to produce reduced time as a learning session occurs. As I noted earlier, among four or ten trials of learning sessions, the predicted task time increased on the second trial. I could not find any reasonable causes for this model behavior.

With respect to forgetting, the ACT-R architecture does not fully support the modeling capabilities. Particularly, there is no theoretically supporting mechanism to model skill decay. The activation values decrease as a power function in declarative memory. However, it is unknown how learned production rules can be unlearned to produce decreased time.

For four serial trials of training ($X1$ to $X4$), the task completion time is getting decreased. Time r with skill disuse after the serial training trials, the task completion time would increase by Δt. Thus, skill decay function can be derived as follows:

$$\Delta t = f(R) - f(X4)$$

$$r + X4 = R$$
$$r > 0$$
$$X_i > 0$$
$$i = 1,2,3,4$$

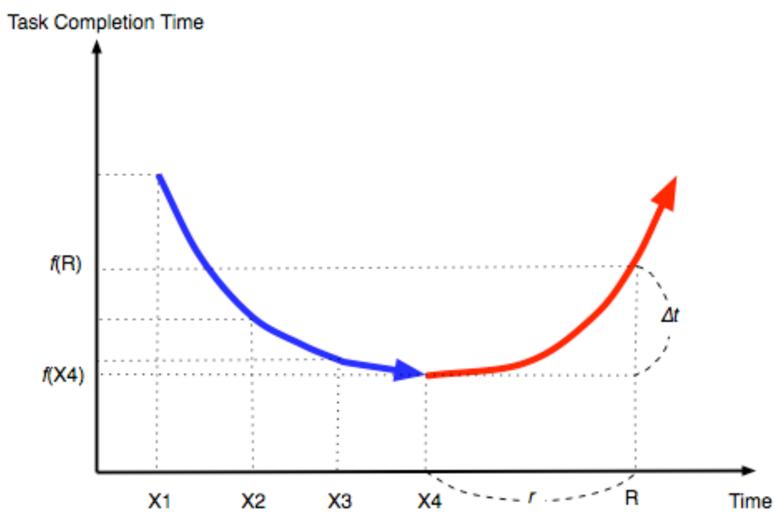


Figure 4.16. A learning and forgetting curve to represent the amount of forgetting after a given retention interval.

If we find out the relationship between r and Δt, it could be possible to predict the amount of skill decay. This mathematical relationship can help to create a new module that predicts skills decay. It is possible to have projected curves of forgetting in time but might still leave an open question that how production rules are unlearned. To answer this "how", it is necessary to produce mechanisms representing skill decay that are psychologically plausible and computationally possible. Instead of finding a regression or fitted curve, these mechanisms can produce forgetting behavior of humans.

4.6 Summary of the Model

The first subtask, open a Dismal spreadsheet file, was modeled in the ACT-R 6 architecture. The model represents a mouse user performing the first subtask. In declarative memory, chunks were created to represent knowledge that was needed for the task. In procedural memory, eleven production rules were written to produce behavior.

As the number of serial trials increases, the model generally showed a decrease in the task completion time, except the second trial. In all simulated trials, it was observed that the second trial produced larger task completion time that the first trial. Intuitively, the first trial should have taken the longest time. Some further investigations on the ACT-R architecture are needed.

When it comes to forgetting, there was no way to decompile learned rules by the production compilation mechanism. The production compilation mechanism combine two rules and creates a new rule, producing faster performance.

Many challenges and problems had been arising during the investigation in this dissertation. Various types of cognitive models under the ACT-R 6 architecture including knowledge acquisition or degradation on a spreadsheet task need to be developed and tested. This will provide a better understanding of user behavior.

Chapter 5
Embodying the Model in an Environment

This chapter describes why we need to embody cognitive models and how to embody them. Embodying the cognitive model can help us to better understand human cognition and compare the model performance with human performance.

5.1 Cognitive Models Fail to Interact with Environments

Cognitive architectures provide a framework upon which to build models that emulate human behavior as discussed before. The model of the user is studied to provide a theoretical and practical understanding of user behaviors and usability of interfaces.

However, there are restrictions placed on the cognitive models regarding access to an external environment. Researchers have studied how to embody a computational cognitive model to interact with a simulated task environment.

Cognitive models generally fail to interact with an external task environment (Ritter, Baxter, Jones, & Young, 2000) although ACT-R/PM (see Byrne, 2001) has helped change this. To enable cognitive models to perform interactive tasks, it is necessary for the models to have visual perception and motor action capabilities. These capabilities allow a cognitive model to perceive what is on the screen and to make some types of mouse movements.

The model's visual perception capabilities should have similar mechanisms to humans. One difference for interactive tasks is that the model's perception in two dimensions is adequate with respect to the interaction with a spreadsheet application in this study.

The model's motor action capabilities should also correspond to the human's motor action mechanisms. A cognitive model would use a mouse or a keyboard when the model interacts with an interface. Possible motor actions include typing a letter, moving a mouse, clicking a mouse button, or moving the eyes, etc.

For example, the Argus system supports an embodied cognitive model interacting with a radar-like target classification task (Gray, 2002; Schoelles & Gray, 2001). In the Argus system, the model and the human subjects use the same interface. It is useful for the development of models including human cognition, human performance or AI agents to have more general access to man-made tasks, task environments, and interfaces, and to have access provided in a cognitively plausible way.

5.2 Cognitive Models with Hands and Eyes

In general, there are two fundamental approaches to provide models with access to a simulated task environment. One approach is to instrument a graphic language such as MCL, Tcl/Tk, Java, or SL-GMS. For example, this approach is taken by ACT-R/PM (Byrne & Anderson, 1998) and simulated hands and eyes models (Ritter, Baxter, Jones, & Young, 2000). These models know what objects to pass from an interface and how to input simulated user commands.

The other approach is to work with the bitmap taken from the screen and parse the screen into objects. This is very robust approach, once done, because all interfaces within the instrumented system become available to the model. For example, SegMan (St. Amant, Riedl, Ritter, & Reifers, 2005) provides a fairly robust approach in that it allows any Windows interface to be seen by models (e.g., ACT-R or occasional Soar models). SegMan, however, has some limitations. It can be somewhat difficult to use and extend. It does not yet recognize all the objects that people do.

5.3 Constructs of the ESEGMAN World

In the dissertation study, the Emacs substrate system, ESEGMAN (Emacs SubstratE: Gate toward MAN-made world) is proposed to help simulate user performance. User performance studies are easy to run with Emacs (including text only web browsing, spreadsheet use, and email use). A model can be connected to the same tasks with a high likelihood of the model being able to see and do the tasks that a user can see and do.

The ESEGMAN world consists of a cognitive model, a substrate, and a simulated spreadsheet task environment. ESEGMAN instruments the graphic interface system of Emacs.

Emacs[8] is an extendable editor that basically functions as an operating system for file editing and related information processing. In the ESEGMAN world, ESSEGMAN embodies a cognitive model interacting with a simulated task. This study opens a possibility of a new cognitive modeling paradigm and extends ACT-R's perception and motor capabilities.

As shown in Figure 5.1, the ESEGMAN world provides an environment where a cognitive model subject and human subjects perform the Dismal spreadsheet task in a laboratory setting as part of a study on learning and forgetting.

The model is built on the ACT-R 6 cognitive architecture. A model agent interacts with the GNU Emacs spreadsheet called Dismal (Ritter & Wood, 2005). For the model to directly interact with the task environment (Dismal), a substrate (ESEGMAN) represented by the eye and hand needs to be implemented. In the real world, humans can directly perform the Dismal spreadsheet tasks. Human performance is recorded by using RUI denoted by Recording User Input (Kukreja, Stevenson, & Ritter, 2006).

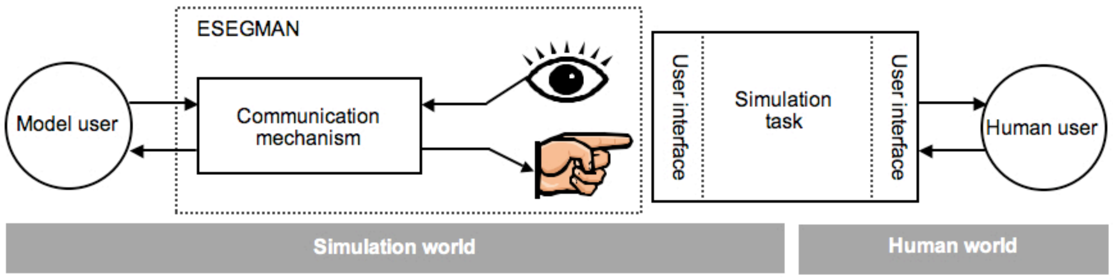


Figure 5.1. A schematic representation of the ESEGMAN world and human world.

5.4 ESEGMAN Mechanisms

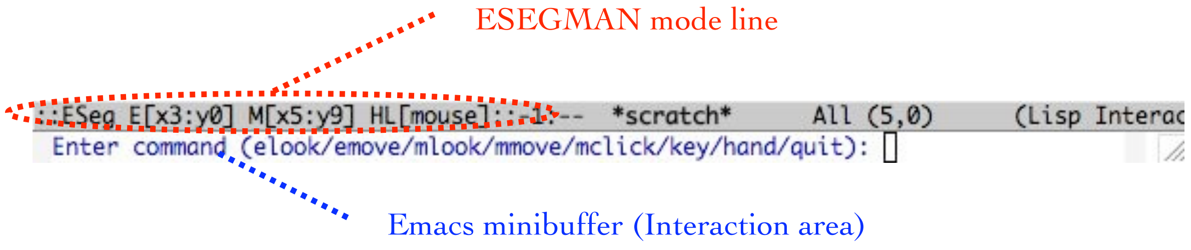
ESEGMAN is layered on the operation of Emacs and allows a model to see and to touch a task environment. The Dismal spreadsheet was implemented in the Emacs Lisp language. ESEGMAN is built on both Common Lisp and Emacs Lisp languages. Thus,

[8] http://www.gnu.org/software/emacs/

ESEGMAN can provide an ACT-R model with a gate to interact with a man-made world of spreadsheet tasks.

ESEGMAN works in the following way. An Emacs shell process is spawned, and a model is loaded within that process. For example, a shell is started in Emacs to invoke OpenMCL that is a Lisp implementation. Then, ACT-R 6 is loaded into OpenMCL. An ACT-R model can send commands to ESEGMAN, such as to move the mouse, to type a letter, or to get the contents of Dismal as fovea. In Emacs, there is a set of functions to take outputs from the shell and insert them into the associated buffer. This approach allows a natural place for ESEGMAN to inspect what is sent, and if a command is sent, execute it.

If the command is to type a letter or to execute a keystroke command, this can be done directly using the extension language of Emacs Lisp. If the command is to move the mouse, a model mouse pointer is moved, shown in the mode line of the buffer being used by the model. If the command is to execute a mouse action, the corresponding process as for keystrokes is executed.



```
::ESeg E[x3:y0] M[x5:y9] HL[mouse]::-1:--  *scratch*      All (5,0)      (Lisp Interac
 Enter command (elook/emove/mlook/mmove/mclick/key/hand/quit): 
```

Figure 5.2. ESEGMAN mode line in the Emacs text editor.

Figure 5.2 shows the mode line of ESEGMAN. In the mode line, you can see several items such as ESeg, E[x3:y0], M[x5:y9], and HL[Mouse]. ESeg indicates the current mode line is ESEGMAN. E[x3:y0], that is a default value, indicates the eye location. The current eye location is (3, 0) in characters. Similarly, the mouse location is represented by M[x5:y9]. The current mouse point location is (5, 9) in characters.

When the model wants to look at the screen, ESEGMAN takes the current fovea location and sets up a data structure to be processed and sends this back to the ACT-R model. ACT-R, after sending the fovea look command, has a read that follows the incoming information and puts it into the ACT-R's visual iconic memory. ESEGMAN can create a file, or it can pass back through the process to an associated buffer.

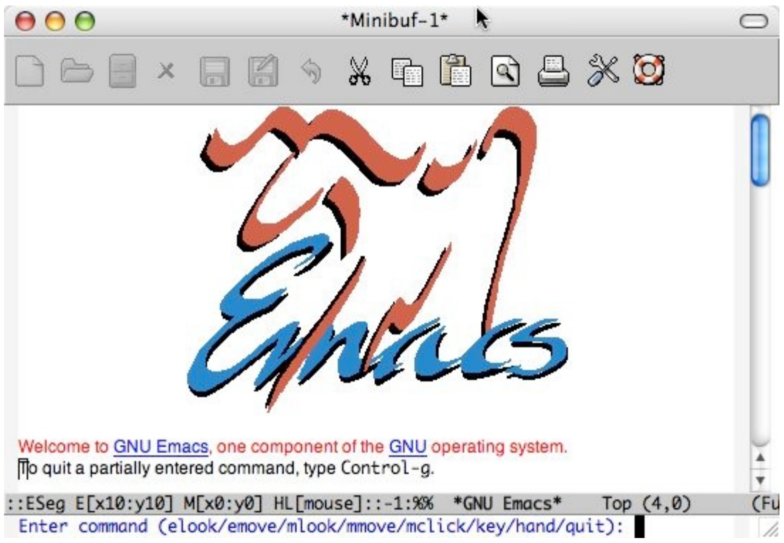
5.5 ESEGMAN Development Notes

I describe the design and development process of ESEGMAN. As noted earlier, ESEGMAN has a simulated-eye that looks at what is on the task screen and gets the visual location of an object on that screen that a model attends to. Also, the simulated-hand for ESEGMAN provides functional operations for key press, mouse move, and mouse click.

5.5.1 Read-Evaluate-Print Loop for ESEGMAN

The command loop in Emacs reads key sequences, executes their definitions, and displays the corresponding results. Similar to this command loop process, ESEGMAN's main function, called *Erepl* (ESEGMAN Read, Evaluate, Print Loop), provides functionality of reading commands, evaluating them, and printing the corresponding results in a target buffer.

Erepl processes commands for the simulated eye and hand, and other utility functions. The commands include "look" for reading visual information on the screen, "emove" for eye movement, "mclick" for mouse click, "mmove" for mouse move, "key" for key press, and "stop" for halting the Erepl process. Commands are found in Figure 4.3.



Welcome to GNU Emacs, one component of the GNU operating system.
To quit a partially entered command, type Control-g.

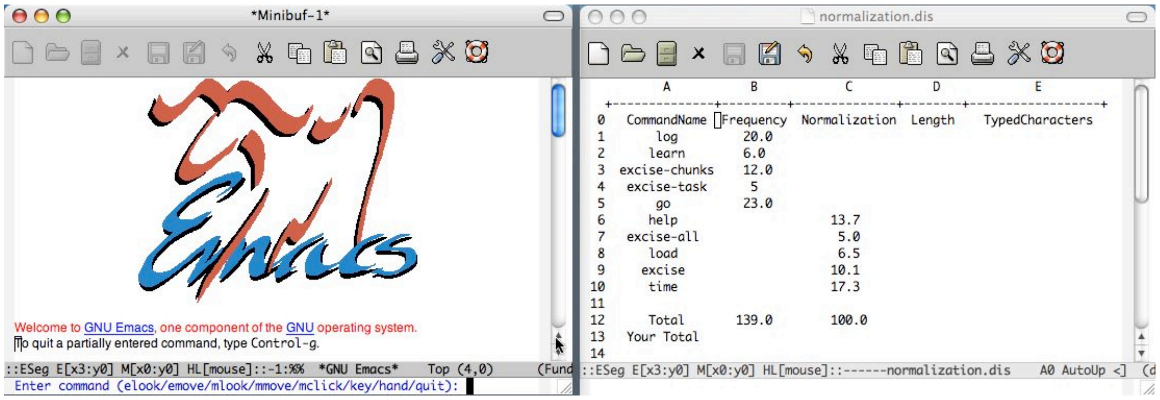
```
::ESeg E[x10:y10] M[x0:y0] HL[mouse]::-1:%%  *GNU Emacs*    Top (4,0)     (Fu
 Enter command (elook/emove/mlook/mmove/mclick/key/hand/quit):
```

Figure 5.3. A picture of Erepl in Emacs. In the minibuffer, several commands are displayed and in the mode line, information about the current eye location of ESEGNAN, the current mouse location, and the current hand location is displayed.

5.5.2 Simulated Eye for ESEGMAN

The simulated eye for ESEGMAN looks at the task window (e.g., the Dismal spreadsheet buffer/window). The function whatis-at takes two arguments of row and column in characters and returns corresponding characters.

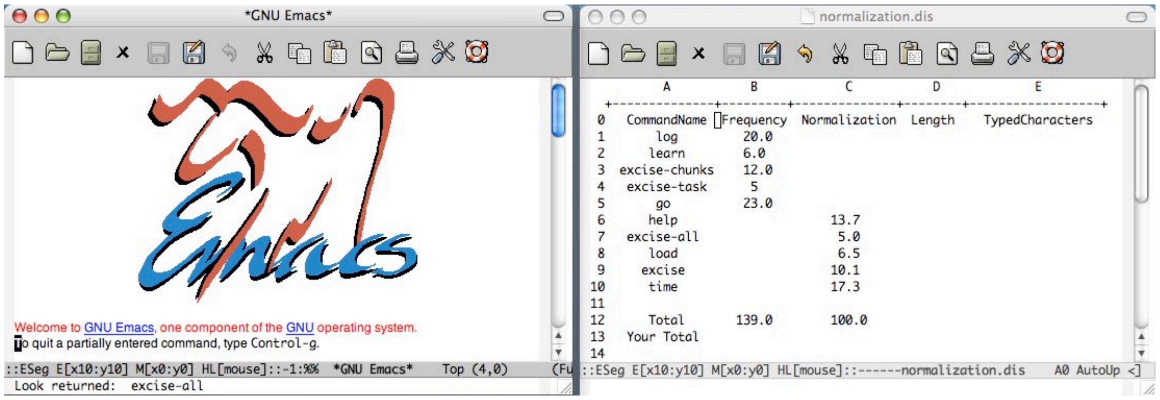
Figure 5.4 shows how Erepl can look at an visual object in the target buffer. A window on the left shows Erepl running in Emacs and commands in the minibuffer. A window on the right is the target window of the Dismal spreadsheet. In Figure 5.4 (a), the current eye location of ESEGMAN is (3, 0) in characters. That is, ESEGMAN's eye is looking at row 3 and the first column. Row 3 indicates the first row of the Dismal spreadsheet task that is labeled 0 in the first column. The eye location is changed to (10, 10), that is ESEGMAN's eye has moved 7 rows down and 10 columns to the right, as seen in Figure 5.4 (b). Finally, ESEGMAN returned excise-all at which ESEGMAN was looking.



```
*Minibuf-1*
Welcome to GNU Emacs, one component of the GNU operating system.
To quit a partially entered command, type Control-g.
::ESeg E[x3:y0] M[x0:y0] HL[mouse]::-1:%%  *GNU Emacs*    Top (4,0)     (Fund
Enter command (elook/emove/mlook/mmove/mclick/key/hand/quit):
```

```
normalization.dis
        A              B            C          D            E
  +--------------+---------+-------------+--------+------------------+
0   CommandName  Frequency  Normalization  Length    TypedCharacters
1       log         20.0
2      learn        6.0
3   excise-chunks   12.0
4    excise-task     5
5        go         23.0
6       help                    13.7
7    excise-all                  5.0
8       load                     6.5
9      excise                   10.1
10      time                    17.3
11
12      Total      139.0       100.0
13   Your Total
14
::ESeg E[x3:y0] M[x0:y0] HL[mouse]::------normalization.dis    A0 AutoUp <]   (d
```

(a) Erepl in Emacs on the left and the target task window right. On the left window, the commands (i.e. elook, emove, mlook, mmove, mclick, key, hand, or quit) are displayed in the minibuffer. The current eye location of ESEGMAN is "E[x3:y0].



```
*GNU Emacs*
Welcome to GNU Emacs, one component of the GNU operating system.
To quit a partially entered command, type Control-g.
::ESeg E[x10:y10] M[x0:y0] HL[mouse]::-1:%%  *GNU Emacs*    Top (4,0)     (Fu
Look returned:  excise-all
```

```
normalization.dis
        A              B            C          D            E
  +--------------+---------+-------------+--------+------------------+
0   CommandName  Frequency  Normalization  Length    TypedCharacters
1       log         20.0
2      learn        6.0
3   excise-chunks   12.0
4    excise-task     5
5        go         23.0
6       help                    13.7
7    excise-all                  5.0
8       load                     6.5
9      excise                   10.1
10      time                    17.3
11
12      Total      139.0       100.0
13   Your Total
14
::ESeg E[x10:y10] M[x0:y0] HL[mouse]::------normalization.dis    A0 AutoUp <]
```

(b) Erepl looking at the task window. The current eye location is updated to E[x10:y10]. The elook command returned "excise-all" from the target window.

Figure 5.4. Erepl in Emacs and the target window of the Dismal spreadsheet.

5.5.3 Simulated Hand for ESEGMAN

The simulated hand for ESEGMAN recognizes mouse movements, mouse clicks, and key presses made by the model subject. The mouse movement mechanism is similar to the mechanism of eye movement as described before. The mode line displays the current location of the mouse.

When a user presses a mouse button and releases it at the same location that generates a mouse click event.

When the model presses a mouse button down and releases it at the same location, ESEGMAN recognized that behavior as a mouse click event. It is assumed that the model (e.g., the ACT-R model) uses a mouse with one button because the ACT-R architecture assumes a virtual mouse with one button.

Special data structures in Emacs, called keymaps, record input events, specifying key bindings for various key sequences. When a user creates an input event (e.g., mouse click) that is bound to a keymap, Emacs finds the next input event by looking up that keymap. This process is called key lookup. Mouse button events cannot be represented as strings, Emacs represents it as a vector.

5.5.4 Other Utility Functions

An Emacs window has a mode line at the bottom, displaying display status information such as the buffer's name, associated file, depth of recursive editing, and major/minor modes. The mode line format is modified to display the buffer status of ESEGMAN. The ESEGMAN mode line displays the eye locations in x and y, the mouse locations in x and y, and the hand location (mouse or keyboard).

5.6 Summary

The ESEGMAN system was designed to embody an ACT-R model, directly interacting with the Emacs text editor and was mostly implemented. Restrictions placed on the cognitive models regarding access to an external environment can be resolved by the ESEGMAN system. Visual perception and motor action capabilities of ESEGMAN can strengthen those capabilities of ACT-R. Currently, the system is not fully working and needs to establish a connection to ACT-R.

Chapter 6
Human Data: Procedural Skills Degradation

In this chapter, I report the study with human participants to explore skills degradation. Participants performed a set of spreadsheet tasks in the study environment that was created for investigating learning and forgetting.

6.1 Method

6.1.1 Participants

Forty-two undergraduate and graduate students at the Pennsylvania State University were recruited to participate in this experiment, including four participants for pilot studies. Five participants could not complete the experiment. Three participants' data were not completely recorded (e.g., RUI stopped recording during the experiment because a participant inadvertently pressed C-s[9]) and two participants dropped out to attend to personal activities (i.e., job interview). Finally, I analyzed data from thirty participants and report all of the results in this section[10]. Participants were paid between $25 and $35 that depended on the number of sessions by different retention intervals.

6.1.2 Materials

As mentioned earlier, the Dismal[11] spreadsheet was implemented to gather and analyze behavioral data (Ritter & Wood, 2005). Dismal is useful here because it is novel to all participants and they do not have any prior experience. A vertical mouse was used to measure participants' motor learning. I assumed that it provides new motor skills to learn (and to forget). The vertical mouse is ergonomically designed to reduce stress on a user's wrist. Instead of a palm-down position of a regular mouse, this vertical mouse

---

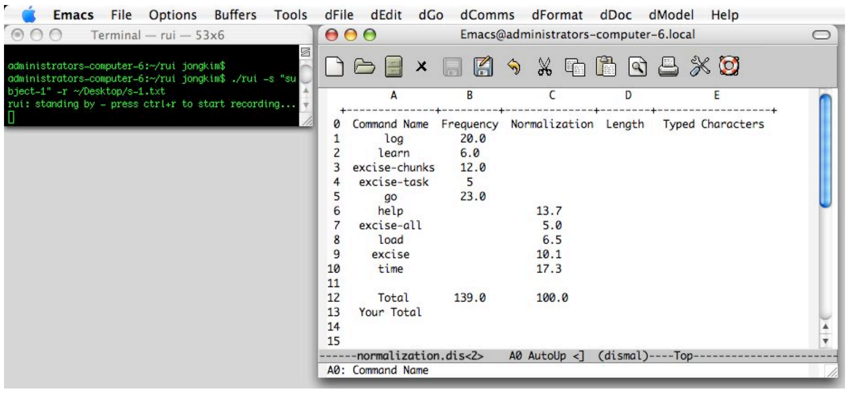
[9] C-s (pressing *s* while holding a control key) is the key command to stop recording in RUI. All participants were informed of not pressing C-s during his/her task.

[10] 19 participants' data were published, see Kim, J. W., Koubek, R. J., & Ritter, F. E. (2007). Investigation of procedural skills degradation from different modalities. In R. L. Lewis, T. A. Polk & J. E. Laird (Eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (pp. 255-260). Oxford, UK: Taylor & Francis/Psychology Press.

[11] http://acs.ist.psu.edu/dismal/dismal.html

requires different hand and forearm postures. None of the participants had prior experience of a vertical mouse and the Dismal spreadsheet, and we could minimize participants' previous knowledge and skills and reduce noise to measure learning effects. Keystrokes, mouse clicks, mouse movements, and task completion time were recorded by the Recording User Input (RUI) system (Kukreja, Stevenson, & Ritter, 2006).

Figure 6.1 shows the study environment with RUI and Dismal. RUI is ready to record inputs from users in an unobtrusive way. In the Dismal spreadsheet, some default values are given in Frequency and Normalization columns. The default values have seven different versions that participants use in each session with different version. Thus, participants worked on the same spreadsheet problems, but the given data were varied.



```
administrators-computer-6:~/rui jongkim$
administrators-computer-6:~/rui jongkim$ ./rui -s "su
bject-1" -r ~/Desktop/s-1.txt
rui: standing by - press ctrl+r to start recording...
```

```
         A              B          C            D          E
   +--------------+---------+-------------+--------+-----------------+
0  Command Name   Frequency  Normalization  Length   Typed Characters
1       log          20.0
2      learn         6.0
3  excise-chunks     12.0
4   excise-task       5
5       go           23.0
6      help                       13.7
7   excise-all                     5.0
8      load                        6.5
9     excise                      10.1
10     time                       17.3
11
12     Total        139.0        100.0
13   Your Total
14
15
-----normalization.dis<2>    A0 AutoUp <]  (dismal)----Top-------------------
A0: Command Name
```

Figure 6.1. The study environment with RUI and Dismal.

As shown In Figure 6.1, the Dismal spreadsheet task consists of five columns (A to E). Column A has ten different names of computer commands. Column B has frequencies of each command listed from row 1 to 5. Column C has normalized frequencies listed in row 6 to 10. There are five blank cells that are filled in by participants, in B and C columns (e.g., B6 to B10, and C1 to C5). Column D and E had ten blank cells that are filled in by participants. The total of the frequency column (row 1 to 10) and the normalization column (row 1 to 10) are provided to the participants. The total of the frequency is used to calculate normalization and frequency.

6.1.3 Design

The experiment is 2 by 3 factorial design with independent variables of modality and retention interval. All participants were randomly assigned to the experimental conditions: (a) three different retention intervals (6-day, 12-day, and 18-day), and (b) two modalities (mouse users and keyboard users). The modality condition consists of two levels including menu-based command users with a vertical mouse (M) and key-based command users with a keyboard (K), representing two different types of skills in the task. The variable of retention interval (R) indicates a period of skills disuse between the last learning (or practice) on Day 4 and the first return day for forgetting measure. The retention interval consists of three levels including 6-day retention interval (R6), 12-day retention interval (R12), and 18-day retention interval (R18). R6 indicates that participants made the first return six days after the last learning. R12 indicates that participants made the first return twelve days after the last learning. R18 indicates that participants made the first return eighteen days after the last learning. During the retention period, participants were asked not to do mental rehearsal or practice of the task.

For the key-based command users (K), fifteen participants performed the procedural spreadsheet task and were not allowed to use a mouse. They were allowed to only use key-based commands with a plain keyboard. For example, to open a file in Dismal, participants need to retrieve a declarative chunk of "`C-x C-f`"[12] and make the relevant key-press.

For the menu-based command users (M), fifteen participants performed the same task using a vertical mouse and were not allowed to use key-based commands. Participants did not get trained to use any skills about the key commands. Participants were only allowed to use mouse-driven menu-based commands. For example, to open a file, they moved the mouse pointer to `File` on the menu bar, then clicked `Open File`.

6.1.4 Procedure

Participants were randomly assigned to groups with regard to modality and retention interval. A learning session was constructed from a study and a test trial. A

[12] `C` indicates holding down the control key while pressing `x`.

forgetting session was constructed only by a test trial. A study trial is when a participant uses the study booklet to learn. Each study task was limited to 30 minutes of study. A test trial is when participants perform the given tasks with the booklet during learning sessions and without the booklet during forgetting sessions.

In the first week, four consecutive learning sessions were held. On Day 1, participants had a maximum of 30 minutes to study the given spreadsheet task and then performed the task. On Days 2 to 4, participants were allowed to refresh their acquired knowledge and skills from Day 1, using the study booklet, and then performed the tasks.

After the four learning sessions in the first week, participants returned for additional trials as part of one of three types of retention interval.

Participants had a 6-, 12-, or 18-day retention interval. For the group with 6-day retention intervals (R6), participants returned back to be measured every 6-days for three times on Day 10, 16, and 22. For the group with a 12-day retention interval (R12), participants returned back to be measured 12 days after the learning session that is on Day 16 and 6 days after the first return, which is on Day 22. For the group with an 18-day retention interval (R18), participants returned back to be measured 18 days after the learning session on Day 22. Table 6.1 describes experimental schedules for learning and forgetting measures.

Table 6.1. Schedules for learning and forgetting experiments

	Sun.	Mon.	Tue.	Wed.	Thur.	Fri.	Sat.
1st Week		Study + Test	Study + Test	Study + Test	Study + Test		
2nd Week				Test			
3rd Week			Test				
4th Week		Test					

Note: In the 1st week (the learning week), participants conducted both study and test sessions. From 2nd to 4th week (the forgetting weeks), participants performed only test sessions.

The overall task consisted of 14 steps (refer to Chapter 2.7). First, they opened a Dismal spreadsheet, saved the file as another name, and completed the complex spreadsheet manipulation by calculating and filling in the blank cells using equations, such as five data normalization calculations, five data frequency calculations, ten calculations of length, ten calculations of total typed characters, four summations of each column, and an insertion of the current date using a Dismal command, *(dis-current-date)*.

6.1.5 Dependent Measures and Data Analysis

The task completion time, as a dependent variable, was recorded in milliseconds by RUI (Recording User Input)[13]. The RUI data provides recording of participants' keystrokes, mouse button clicks (pressed and released), and mouse movements (e.g., xy coordinates of mouse locations in pixels). RUI assumed that the mouse has one button, like ACT-R's assumption.

The RUI output is a text file that can be imported to any spreadsheet or statistics applications. I import the RUI data into Excel and SPSS for analysis. The size of the RUI data from each trial ranges from 16 to 284 kilobytes.

[13] available from acs.ist.psu.edu/rui/

6.2 Results and Discussion

I report all of the data from thirty participants here. In the General Performance section, data of participants' task completion time were analyzed to investigate how well participants learned the task and how much they forgot the task in terms of modality and retention interval. In the R6 group, ten participants completed the task (5 for mouse users and 5 for keyboard users). The group of R12 and R18 is also comprised of 10 participants respectively (5 for mouse users and 5 for keyboard users).

6.2.1 General Performance of Learning

All of the thirty-participants completed the learning sessions. Figure 6.2 shows the average time with respect to the two modalities over the four consecutive days of learning. The average task completion time for keyboard users (K) decreased from 1,532 (±284) to 631 (±116) s. The average task completion time for mouse users (M) decreased from 1392 (±356) s to 697 (±119) s. The learning curves of the mouse and keyboard groups follow the Power law of learning:

$$y = 1498.8x^{-0.6317},\ R^2 = 0.99 \text{ for the keyboard group}$$

$$y = 1348.6x^{-0.5071},\ R^2 = 0.97 \text{ for the mouse group}$$

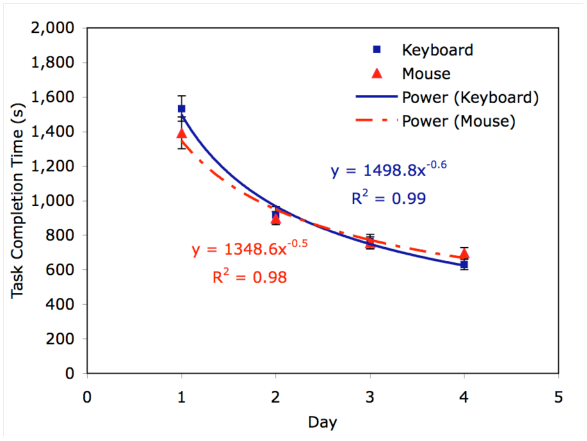


Figure 6.2. The Power curves of learning for the groups of keyboard vs. mouse.

Independent samples *t*-tests were conducted for mouse and keyboard users for each study session. I assumed that two samples are independent and are normally distributed. Also, it is assumed that populations have equal variance. There were no significant differences, for all comparisons, $t(28) < 1.55$, $p >= .13$. These results suggest that the input/manipulation style factor, keystroke or mouse driven, did not lead to significant differences in learning on this task over this time range and for this population. Figure 6.3 shows the log-log plot of learning curves of the two-modality groups (keyboard vs. mouse). This figure indicates that the groups all learned over the four learning sessions. They all performed at pretty much the same level. Similar to the famous log-log linear plot of skill acquisition (Newell & Rosenbloom, 1981), the investigation of the spreadsheet skill acquisition, here, also confirmed a linear relationship between performance and practice.

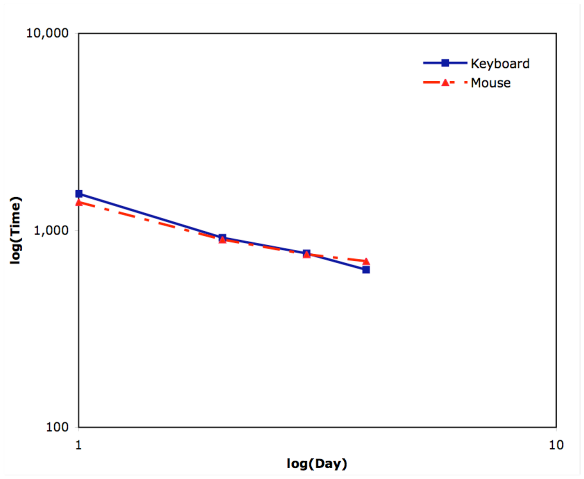


Figure 6.3. The log-log plot of learning curves for keyboard and mouse users.

6.2.2 General Performance of Forgetting

Overall Performance from Day 1 to Day 22

The R6 groups (keyboard and mouse users) had a 6-day retention for their first return and two additional six-day retention intervals. The R12 groups of keyboard and mouse users had 12-day retention for the first return, and an additional 6-day retention.

The R18 groups of keyboard and mouse users had one 18-day retention for the first return and no additional return.

Figure 6.4 shows task performance of participants with a 6-day retention interval as a first return. The 6-day retention caused an increase in task completion time on Day 10. The average time on Day 10 is 716 s (±168) for the keyboard group and 929 s (±251) for the mouse group. The additional 6-day retention after the 6-day retention produced decrease in task completion time, showing that the 6-day retention can serve as a distributed learning. The task completion time on Day 16 is 640 s (±177) for the keyboard group and 585 s (±90) for the mouse group. The average time on Day 22 is 575 s (±118) for the keyboard group and 686 s (±242) for the mouse group.

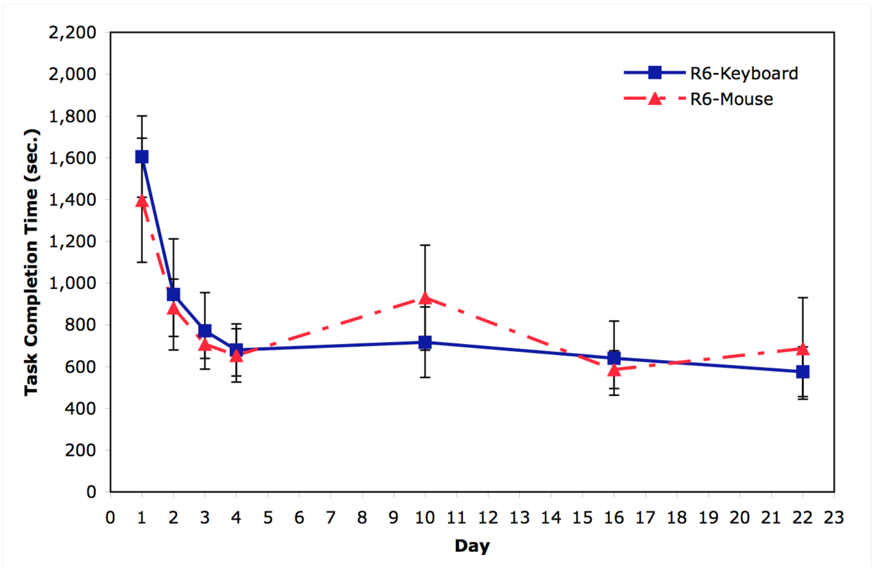


Figure 6.4. Learning and forgetting curves with standard error bars of two modalities, with a 6-day retention interval as a first return and two additional six-day retention intervals.

Figure 6.5 shows task performance of participants with a 12-day retention interval as a first return. The 12-day retention caused an increase in task completion time on Day 10. The average time on Day 10 is 883 s (± 344) for the keyboard group and 878 s (±156) for the mouse group. The 6-day retention after the 12-day retention produced decrease in task completion time, showing that the 6-day retention can serve as a distributed learning. The average time on Day 22 is 598 s (±182) for the keyboard group and 654 s (±76) for the mouse group. This task completion time on Day 22 approximately approached to the time on Day 4 (589 s for keyboard and 672 s for mouse).

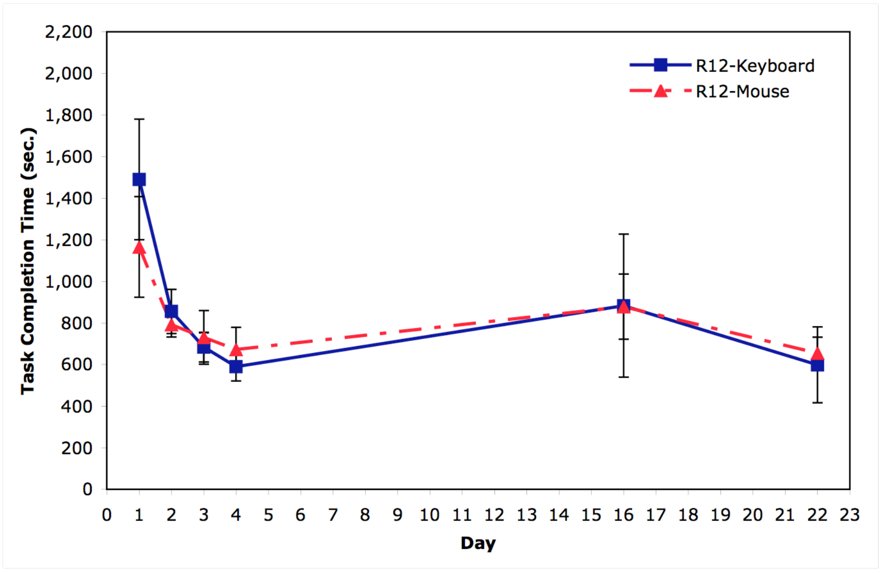


Figure 6.5. Learning and forgetting curves with standard error bars of two modalities, with 12-day retention interval as a first return.

Figure 6.6 shows task performance of participants with an 18-day retention interval as a first return. The task completion time on Day 4 is 625 s (±148) for the keyboard group and 768 s (±111) for the mouse group. The keyboard group showed faster performance than that of the mouse group by approximate difference of 143 s. Then, after the 18-day retention, the task completion time on Day 22 is 1371 s (±328) and 1222 s (±184). The keyboard group approximately took 149 s more than that of the mouse group.

This result should remain tentative because of the small sample size here. But, it gives somewhat interesting notes: (a) learning motor skills, that is participants who have not used the vertical mouse before learned to use it for the task, can slow performance of users, and (b) once new motor skills are acquired, they can be less susceptible to decay than declarative knowledge retrievals (e.g., keyboard users use key-based commands but mouse users use menu-driven commands).

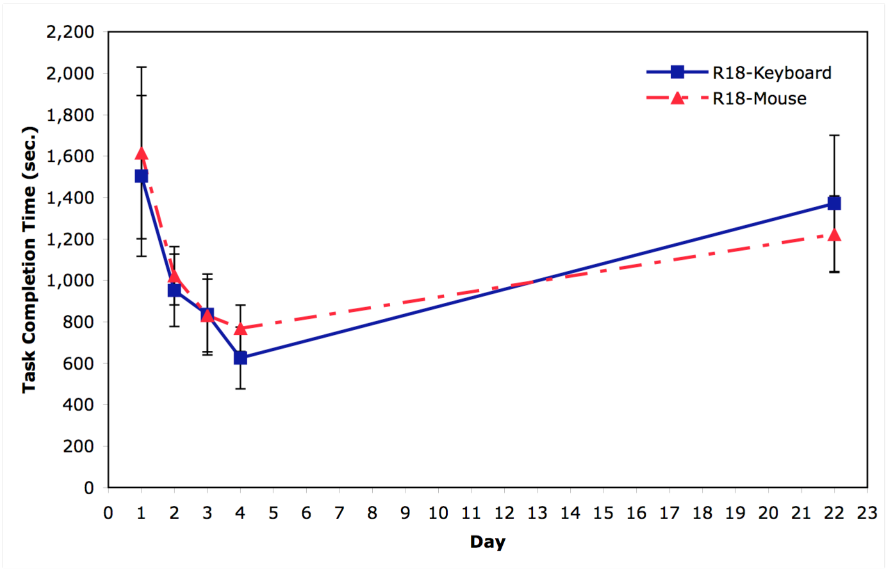


Figure 6.6. Learning and forgetting curves with standard error bars of two modalities, with 18-day retention interval as a first return.

Performance on Day 4 and the First Return Day

Participants in the keyboard group (n = 15) completed the task in 631 s and ones in the mouse group (n = 15) completed the task in 697 s on Day 4, shown in Table 6.2 There is one minute delay on the task completion time between two groups of mouse and keyboard users. It was found that there was no sufficient evidence of the difference in task completion time on Day 4, $t(28) = -1.55$, $p > .05$.

An interesting question arises from the data: why does the menu-based environment not help users to better learn the task? I presumed that the keyboard that participants used in this experiment is not novel. Participants actually started learning to use the keyboard before the onset of the experiment. In the meanwhile, the novel vertical mouse required participants to learn a new type of motor skills. It can be presumed that the motor skill acquisition could nullify the benefits from the learning-friendly environment of the menu driven task.

Table 6.3 shows all participants' task completion time in term of retention intervals and modality. It is observed that the 6K group on the First Return has lower task completion time than the one of the 6M group, 716 s for the keyboard and 929 s for the mouse group (around 213 s difference). The average task completion time of the 12K group is 883 s and that of the 12M group is 878 s, indicating similar performance, around 4 s difference. The average task completion time of the 18K group is 1371 s and that of the 18M is 1223 s, indicating around 148 s difference.

More forgetting was observed in the mouse group with the short retention interval (6-day). In the intermediate retention interval (12-day), similar forgetting performance between the different modality groups was observed. In the long retention interval (18-day), more forgetting in the keyboard group was observed.

There is no statistically significant difference in the task completion time on the first return day, $t(8) = -1.58$, $p > .05$ for the 6-day retention interval, $t(8) = 0.03$, $p > .05$ for the 12-day retention interval, and $t(8) = 0.88$, $p > .05$ for the 18-day retention interval.

Table 6.2. Descriptive statistics on Day 4.

R	Modality	*N*	Mean	*SD*	*SE*	*t*	*p*
6	K	5	679.63	124.39	55.63		
	M	5	653.54	127.25	56.91	*t*(8) = 0.33	0.75
	Total	10	666.59	119.43	37.77		
12	K	5	589.34	68.23	30.51		
	M	5	672.09	106.69	47.71	*t*(8) = -1.46	0.18
	Total	10	630.72	95.03	30.05		
18	K	5	625.38	148.83	66.56		
	M	5	768.33	111.92	50.05	*t*(8) = -1.73	0.12
	Total	10	696.86	145.22	45.92		
Total	K	15	631.45	116.43	30.06		
	M	15	697.99	119.04	30.74	*t*(28) = -1.55	0.13
	Total	30	664.72	120.54	22.01		

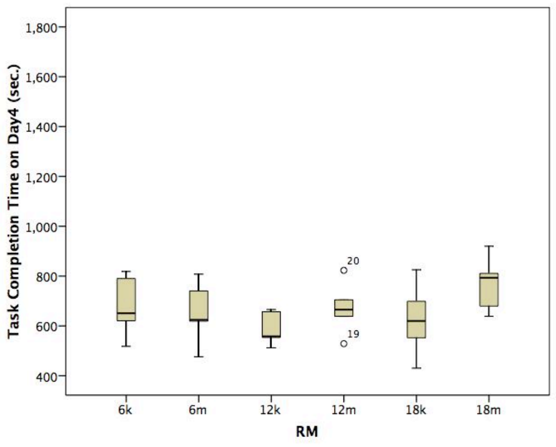
Note: *SE* indicates Standard Error of Mean.

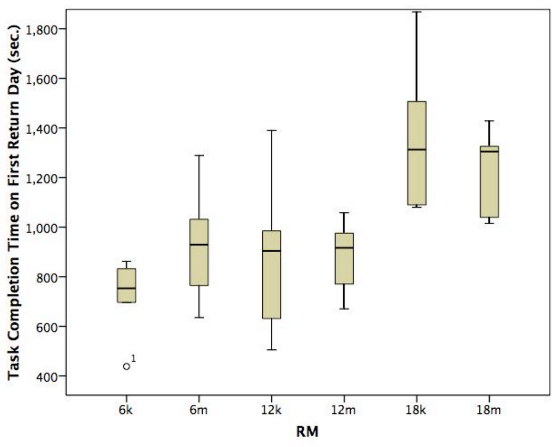
Table 6.3. Descriptive statistics on the First Return Day.

R	Modality	*N*	Mean	*SD*	*SE*	*t*	*p*
6	K	5	716.39	168.60	75.40	*t*(8) = -1.58	0.15
	M	5	929.89	251.66	112.55		
12	K	5	883.11	344.25	153.96	*t*(8) = 0.03	0.98
	M	5	878.22	156.46	69.97		
18	K	5	1371.33	328.80	147.05	*t*(8) = 0.88	0.40
	M	5	1222.66	184.60	82.56		

Note: *SE* indicates Standard Error of Mean.

The Boxplots in Figure 6.7 show task performance on Day 4 and the first return day in terms of retention interval and modality. On the x-axis, RM is a combined variable, indicating both retention interval and modality. For example, 6K indicates participants using a keyboard with 6-day retention interval and 6M indicates participants using a vertical mouse with 6-day retention interval. There are individual differences and also outliers in the 12M group on Day 4. As mentioned earlier, there is no modality effect on learning performance with the sample size ($N = 30$), leaving a suspicion of modality difference.





(a) Day 4 vs. RM (b) First Return vs. RM

Figure 6.7. Boxplots showing task completion time (in sec.) on Day 4 and the first return by retention interval and modality factors.

Modality and Retention Interval on Forgetting

The task completion time on Day 4 can indicate the degree of learning. That is, the time on Day 4 plays a role as an indicator of how well participants have learned the given task. Therefore, I included the time on Day 4 as a covariate into the model below because the degree of learning can affect forgetting performance. R (three types of retention intervals) and modality (keyboard and vertical mouse users) are fixed factors.

$$\text{First Return} = \text{Day4} + \text{R} + \text{Modality} + \text{R*Modality}$$

Table 6.4 shows the results of ANOVA for the task performance on the first return. The main effect of retention interval is significant, $F(2, 27) = 9.96$, $p < .05$. The main effect of modality is not significant, *F*(1, 28) = 0.06, *p* > .05. There is no significant interaction effect between retention interval (R) and modality, *F*(2, 27) = 1.07, *p* > .05. The covariate effect (Day 4) is not significant, *F*(1, 28) = 0.02, *p* > .05.

Table 6.4. ANOVA table for task performance on the first return. The model is task time on First Return = Day4 + R + Modality + R*Modality.

Source	*df*	*Sum of Squares*	*Mean Square*	*F*	*p-value*
Model	6	1,508,069	251,344	3.84	0.008
Intercept	1	795,225	795,225	12.14	0.002
D4	1	1,517	1,517	0.02	0.880
R	2	1,304,141	652,070	9.96	0.001*
Modality	1	4,087	4,087	0.06	0.805
R*Modality	2	140,274	70,137	1.07	0.359
Error	23	1,506,240	65,488		
Total	30	33,030,267			
Corrected Total	29	3,014,309			

Now, let us determine which group of the mean task completion time is different from each other. I conducted multiple comparisons of the task completion time on the first return day with respect to retention interval shown in Table 6.5.

The 6-day (R6) and 12-day (R12) retention intervals did not have a statistically significant difference with each other on the task completion time, *t*(18) = -0.53, *p* > .05. The 6-day (R6) and 18-day (R18) had significant difference on the task completion time, *t*(18) = -4.28, *p* < .05. Also, the 12-day (R12) and 18-day (R18) had significant difference on the task completion time, *t*(18) = -3.61, *p* < .05.

Table 6.5. Pairwise comparisons of the task completion time on the first return day with respect to three different retention intervals (R6, R12, and R18).

R		Mean Diff.	*SE*	*n*	*t*(18)	*p*	95% CI for difference	
							LB	UB
R6	R12	-101.35	131.79	20	-0.77	0.45	-378.23	175.53
R6	R18	-517.68	139.64	20	-4.86	0.00*	-799.13	-236.23
R12	R18	-416.33	115.28	20	-3.61	0.00*	-658.52	-174.14

Note: *SE* indicates Standard Error of Mean Difference.

I also conducted multiple comparisons of the task completion time on the first return day in terms of each combination of both modality and retention interval. The comparison tests whether the difference between two combinations of modality and retention interval is significant. Table 6.6 shows the mean task completion time on the first return day with respect to both modality and retention interval. Table 6.7 shows pairwise comparisons of the mean task completion time on the first return day.

Table 6.6. Mean task completion time (s) with standard deviation (*SD*) and standard error of mean (*SE*) by modality and retention interval on the first return day.

R	Modality	*n*	First Return Day	Mean	*SD*	*SE*
R6	Keyboard	5	Day 10	716.39	168.60	75.40
	Mouse	5	Day 10	929.89	251.66	112.55
R12	Keyboard	5	Day 16	883.11	344.26	153.96
	Mouse	5	Day 16	878.22	156.46	69.27
R18	Keyboard	5	Day 22	1,371.33	328.80	147.05
	Mouse	5	Day 22	1,222.66	184.60	82.56

Note: *SE* indicates Standard Error of Mean.

For keyboard users, the 6-day (R6) and 18-day (R18) retention intervals resulted in significant differences on the task performance, $t(8) = -3.96$, $p < .05$. The 12-day (R12) and 18-day (R18) retention intervals had significant differences as well, $t(8) = -2.29$, $p = .05$. However, the 6-day (R6) and 12-day (R12) retention intervals did not show any significant differences of task performance, $t(8) = -0.97$, $p > .05$.

For mouse users, there were no significant differences in the task completion time on R6 vs. R12, $t(8) = 0.39$, $p > .05$ and R6 vs. R18, $t(8) = -2.10$, $p > .05$. The task completion time on R12 vs. R18 produced significant differences, $t(8) = -3.18$, $p < .05$.

Table 6.7. Pairwise comparisons of the mean responses on the first return day.

R	Modality	*n*	*df*	Mean Diff.	*SE*	*t*	*p*
R6 vs. R12	Keyboard	10	8	-166.72	171.43	-0.97	0.36
R6 vs. R18		10	8	-654.95	165.25	-3.96	0.00*
R12 vs. R18		10	8	-488.22	212.90	-2.29	0.05*
R6 vs. R12	Mouse	10	8	51.67	131.52	0.39	0.71
R6 vs. R18		10	8	-292.77	139.58	-2.10	0.07
R12 vs. R18		10	8	-344.40	108.22	-3.18	0.01*

Note: *t*(8, 0.05) = 2.31. *SE* indicates Standard Error of Mean Difference.

What Forgetting Curves Look Like?

To fit the forgetting curves, I simply approached to use the data on the first return day (i.e., Day 10 for the R6 group, Day 16 for the R12 group, and Day 22 for the R18 group). I also considered the task completion time on the last training day because the value on the first return does not have any information on the previous learning, indicating how much participants forgot from the learning.

A column of a variable, Y*, was generated to indicate the difference between the task completion time on the first return (First_Return) and the last learning on Day 4. The "First_Return" includes the task completion time on Day 10, Day 16, or Day 22. The variable, Y*, is a linear transformation to preserve the shape of the fitted forgetting curve. That is, the task completion time at Day 4 serves as a baseline from which I am able to measure the quantitative amount of forgetting.

There were a couple of values of Y* being negative (3 out of 15 participants in the keyboard group, and 1 out of 15 participant in the mouse group). This indicates that some participants continue to learn after the retention interval. It was observed that S9, S26, and S34 from the keyboard users continued to learn, producing negative values, shown in Table 6.8. It was also observed that S17 from the group of mouse users continued to learn, producing negative values, shown in Table 6.9.

A negative value makes it impossible to fit exponential and power models because logarithms must be taken to linearize the models. To address this problem, I considered a participant who continues to learn as someone whose forgetting is null. I assigned a very small number (i.e., Y* = 0.00001) by replacing the negative, as seen in Table 6.8 and Table 6.9.

Table 6.8. Task completion time (in second) for keyboard users at three retention intervals (R) of 6-, 12-, and 18-days

Participant	Modality	R	Day 4	First Return	Y*
S9	K	6	518.02	438.23	0.00001
S15	K	6	650.84	696.49	45.65
S16	K	6	620.74	861.93	241.22
S26	K	6	790.11	753.18	0.00001
S31	K	6	818.45	832.11	13.66
S11	K	12	553.53	1,389.98	836.45
S27	K	12	665.77	904.04	238.27
S32	K	12	512.20	631.55	119.35
S34	K	12	558.21	504.93	0.00001
S40	K	12	657.01	985.05	328.04
S12	K	18	552.44	1,079.83	527.39
S14	K	18	619.89	1,867.81	1,247.93
S19	K	18	825.27	1,312.73	487.47
S20	K	18	698.67	1,089.77	391.11
S23	K	18	430.64	1,506.51	1,075.88

Note: In the table, Y* indicates the difference between the task completion time at Day 4 and the one at the first return day or 0.00001, if the difference is negative.

Table 6.9. Task completion time (in second) for mouse users at three retention intervals (R) of 6-, 12-, and 18-days.

Participants	Modality	R	Day 4	First Return	Y*
S7	M	6	624.41	635.25	10.85
S17	M	6	807.67	764.61	0.00001
S29	M	6	476.24	1,288.86	812.62
S30	M	6	619.02	1,031.59	412.57
S33	M	6	740.37	929.12	188.75
S5	M	12	665.36	1,057.97	392.61
S8	M	12	638.56	770.44	131.88
S35	M	12	704.63	975.51	270.88
S37	M	12	528.85	670.69	141.84
S38	M	12	823.04	916.50	93.46
S13	M	18	810.63	1,428.24	617.62
S18	M	18	638.50	1,325.70	687.20
S21	M	18	793.25	1,304.88	511.63
S24	M	18	919.81	1,015.10	95.30
S39	M	18	679.48	1,039.38	359.90

Note: In the table, Y* indicates the difference between the task completion time at Day 4 and the one at the first return day, or 0.00001 if the difference is negative.

First, I looked at the performance at Day 4 and First Return with 6-, 12-, or 18-day retention interval. The reason is that the data on Day 4 provides information on how well participants learned the task and the data on the first return provides information on how much acquired skills are forgotten.

As seen in Figure 6.8, it is observed that participants in R18 produces higher values of task completion time on the first return, followed by R12, and R6 for those who used mouse and for those who used keyboard.

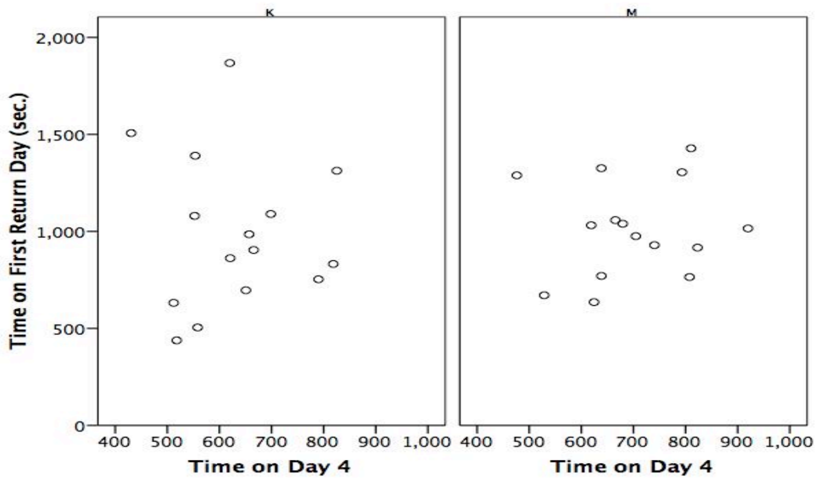


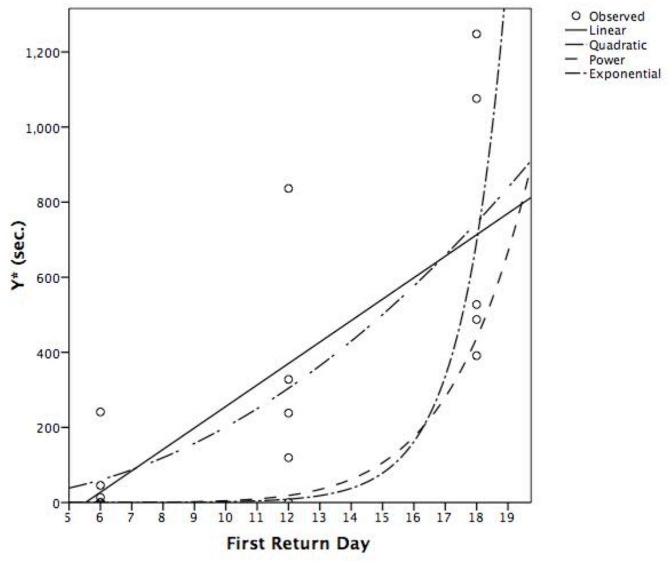
Figure 6.8. Scatter plots of task completion time (in sec.) on the first return made by both keyboard (K) and mouse (M) users. The numbers on the right side of points show retention intervals (6, 12, or 18).

Figure 6.9 and Figure 6.10 show estimated curves based on the observed data. Here, I considered power, exponential, linear, and quadratic models to estimate the curve. However, the quadratic model has the highest R^2, but it is not realistic to compare the quadratic model to other models because of the different number of predictors. The quadratic model does not necessarily demonstrate a superior model.

For the curve estimation of keyboard users, the linear model (p = .002) shows statistical significance on the fitted curve. Other power and exponential models exhibited similar performance and both models are marginally significant (p <= .05), shown in Table 6.10.

For the curve estimation of mouse users, the power model has the smallest p-value (p = .13), shown in Table 6.11. It is unfortunate that all models are insignificant.

There were large individual differences leading to insignificant curve estimation among the models. The result here should remain tentative. The forgetting trends across the different retention intervals were recognizable within participants.



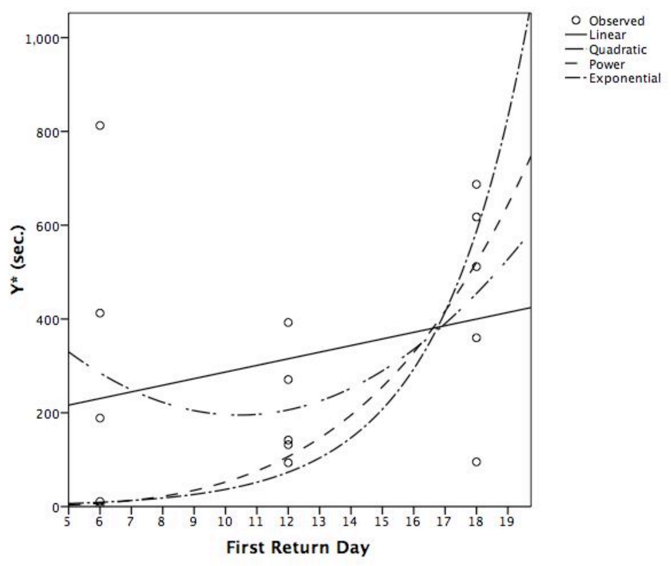
Power: $Y^* = 7E-08x^{7.7843}$, $R^2 = 0.26$
Exponential: $Y^* = 0.0014e^{0.7274x}$, $R^2 = 0.26$
Linear: $Y^* = 57.154x - 315.69$, $R^2 = 0.52$
Quadratic: $Y^* = 2.739x^2 - 8.584x + 13$, $R^2 = 0.53$

Figure 6.9. Fitted forgetting curves for keyboard users.

Table 6.10. Statistical output of curve estimation (keyboard users).

Model	R^2	*df*	*F*	*p*
Linear	0.52	13	14.01	0.00
Quadratic	0.53	12	6.85	0.01
Power	0.26	13	4.54	0.05
Exponential	0.26	13	4.65	0.05

Note: The dependent variable is Y* and the independent variable is Return Day (6, 12, or 18).



Power: $Y^* = 0.0064x^{3.914}$, $R^2 = 0.17$
Exponential: $Y^* = 1.1432e^{0.3467x}$, $R^2 = 0.15$
Linear: $Y^* = 14.114x + 145.77$, $R^2 = 0.08$
Quadratic: $Y^* = 4.542x^2 - 94.893x + 690.8$, $R^2 = 0.18$

Figure 6.10. Fitted forgetting curves for mouse users.

Table 6.11. Statistical output of curve estimation (mouse users).

Model	R^2	*Df*	*F*	*p*
Linear	.08	13	1.12	.31
Quadratic	.18	12	1.30	.31
Power	.17	13	2.59	.13
Exponential	.15	13	2.33	.15

Note: The dependent variable is Y* and the independent variable is Return Day (6, 12, or 18).

6.2.3 Relearning

As mentioned before, participants completed serial training sessions for four days, and then they were made multiple returns for a test. I presumed that the test on the first return day, which is designed to measure forgetting, is able to serve as a relearning of the task for the participants as a side effect. Thus, analysis on these additional tests after the first return can provide an understanding of the relearning effect in terms of two modalities.

Participants in the R6 group made the first return on Day 10 after the serial training sessions. Six days after this first return, all participants including mouse and keyboard users made a return for an additional test on Day 16. Table 5.13 and Table 5.14 show descriptive statistics and test statistics of Day 10 and Day 16. The task completion time of the R6-Keyboard group was reduced by 76 s (from 716 s ± 169 to 640 s ± 178). The task completion time of the R6-Mouse group was reduced by 345 s (from 930 s ± 252 to 585 s ± 90).

I conducted Paired-Samples Test to compare the mean task completion time on Day 10 and Day 16. For mouse users, there is a significant difference of the mean task completion time on Day 10 and Day 16, $t(4) = 3.39$, $p = .03$. For keyboard users, there is no significant evidence of the differential relearning effects, $t(4) = 2.03$, $p > .05$.

Table 6.12. The mean task completion time (s) with standard deviation and standard error of mean on Day 10 and Day 16.

Modality	Day	*N*	Mean	*SD*	*SE*
Keyboard	Day 10	5	716.39	168.60	75.40
	Day 16	5	640.17	177.51	79.38
Mouse	Day 10	5	929.89	251.66	112.55
	Day 16	5	585.39	90.12	40.30

Table 6.13. Comparison of the mean difference and the paired-samples test statistics on Day 10 and Day 16 of keyboard and mouse users.

Modality	Day		Mean	*SD*	*SE*	*t*(4)	*p*
Keyboard	Day 10	Day16	76.22	84.14	37.63	2.03	0.11
Mouse	Day 10	Day 16	344.50	227.28	101.64	3.39	0.03*

Participants in the R12 group made the first return on Day 16 after the serial training sessions. Six days after this first return, all participants including mouse and keyboard users made a return for an additional test on Day 22. Table 5.15 and Table 5.16 show descriptive statistics and test statistics of Day 16 and Day 22.

The task completion time of the R12-Keyboard group was reduced by 285 s (from 883 s ± 344 to 598 s ± 182). The task completion time of the R12-Mouse group was reduced by 224 s (from 878 s ±156 to 655 s ± 77).

I conducted the paired-samples t-test to compare the mean task completion time on Day 16 and Day 22, as well. For mouse users, there is statistical significance of difference the mean task time on Day 16 and Day 22, $t(4) = 3.32$, $p < .05$. For keyboard users, I have to embrace there is no statistical difference by the relearning, $t(4) = 2.60$, $p > .05$. However, the *p*-value is very close to the reject area, leaving the need for a further investigation to find differences. Figure 6.11 shows the plots of the relearning by different modalities and retention intervals.

Table 6.14. The mean task completion time (s) with standard deviation and standard error of mean on Day 16 and Day 22.

Modality	Day	*N*	Mean	*SD*	*SE*
Keyboard	Day 16	5	883.11	344.26	153.96
	Day 22	5	598.63	182.46	81.60
Mouse	Day 16	5	878.22	156.46	69.97
	Day 22	5	654.63	76.96	34.42

Table 6.15. Comparison of the mean difference and the paired-samples test statistics on Day 16 and Day 22 of keyboard and mouse users.

Modality	Day		Mean	*SD*	*SE*	*t*(4)	*p*
K	Day 16	Day 22	284.48	244.54	109.36	2.60	0.06
M	Day 16	Day 22	223.59	150.46	67.29	3.32	0.03*

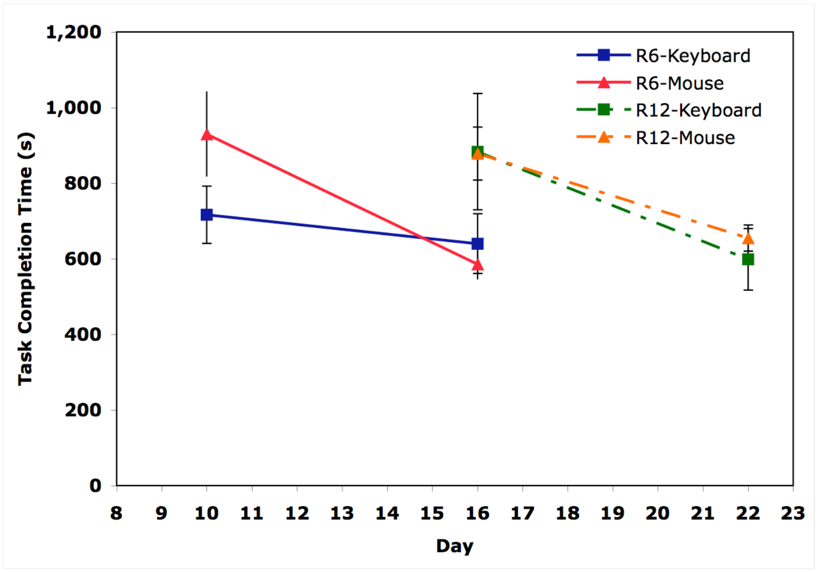
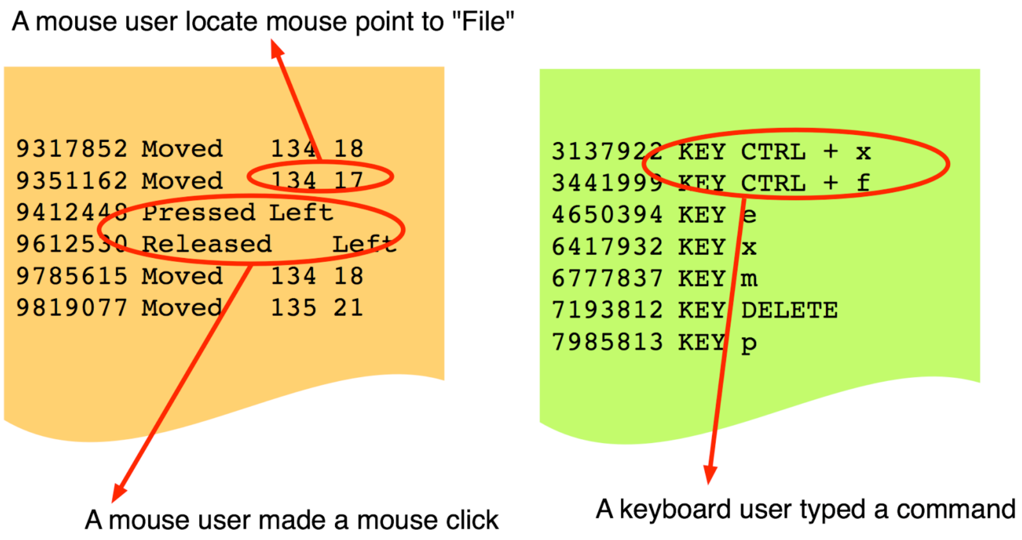


Figure 6.11. The relearning effects of the four groups (R6-Keyboard, R6-Mouse, R12-Keyboard, and R12-Mouse).

6.3 Subtask Analysis

As mentioned earlier, the Dismal spreadsheet task consists of 14 subtasks. Each subtask has different attributes of knowledge. For example, the subtask of filling in the normalization column requires more cognitive problem solving capability from participants than inserting two rows in the spreadsheet. Thus, analysis of subtasks provides deeper understanding of knowledge and skills attributes. Here, I only report the analysis of the first subtask to test the model and its theory that were discussed in Chapter 3. The subtask analysis can provide important benefits to understanding human cognition and learning and to validate accuracy of general performance analysis. That is, I can examine whether the general performance analysis was correct or not, by looking inside the data. I can find out whether participants' performance of the Dismal spreadsheet task is comprised of 14 subtasks. Also I can find out whether participants' performed what they learned to complete each subtask. For example, I can examine if a participant retrieves a correct equation for his/her frequency calculation or normalization calculation. The subtask analysis helps me acknowledge that there might be data that I must throw away or that I should keep. Furthermore, each subtask was designed to represent different knowledge and skills. By comparing each subtask’s learning and forgetting results, it is promising to better understand human cognition.

Raw data gathered from RUI have two types: (a) data from mouse users and (b) data from keyboard users. The RUI data of mouse users consists of activities such as keystrokes, mouse moves, and mouse clicks. The RUI data of keyboard users consists of keystrokes only. Figure 6.12 shows how RUI data look like.


A mouse user locate mouse point to "File"
A mouse user made a mouse click
A keyboard user typed a command

```
9317852 Moved    134 18
9351162 Moved    134 17
9412448 Pressed Left
9612530 Released    Left
9785615 Moved    134 18
9819077 Moved    135 21
```

```
3137922 KEY CTRL + x
3441999 KEY CTRL + f
4650394 KEY e
6417932 KEY x
6777837 KEY m
7193812 KEY DELETE
7985813 KEY p
```

Figure 6.12. Raw data from RUI for a subtask analysis.

To analyze RUI data from mouse users, it is necessary to identify the location of visual objects in pixels that users look at. Table 6.16 gives locations in pixels of visual objects that participants look at to perform the subtask 1, “OPEN FILE”, and the subtask 2, “CALCULATE FREQUENCY”. The pixel information of the objects serves as an important criterion to divide each subtask among the whole 14 subtasks because the user makes a mouse click onto the visual object during the performance. RUI records this mouse click. Based on these mouse clicks made by users, I visually examined the RUI output to extract each subtask and its timing information but this could be automated.

Table 6.16. Locations of visual objects (e.g., menus, a folder, or a file) that users look at and make a click on.

Visual objects	X		Y	
	Lower	Upper	Lower	Upper
Subtask 1: OPEN FILE				
`FILE`	108	150	0	24
`OPEN FILE`	108	366	48	60
`Experiment`	966	1044	564	576
`Normalization.dis`	942	1068	468	480
`CHOOSE`	1020	1092	732	750
Subtask 3: CALCULATE FREQUENCY				
`Cell B6`	678	744	366	378
`Cell B7`	678	744	378	396
`Cell B8`	678	744	396	414
`Cell B9`	678	744	414	426
`Cell B10`	678	744	426	438
`dEdit`	396	450	0	24
`Edit cell (E)`	396	630	102	114

Note: The unit of the value is pixel. The screen that participants used in the experiment is Apple Cinema HD Display 23″ (1680*1050 resolution).

I plotted the learning performance of the subtask 1, OPEN FILE, from the 15 mouse users. Table 6.17 provides descriptive statistics on the learning performance. As seen in Figure 6.13, the learning curve fits to a Power function, $y = 24.4x^{-0.45}$, $R^2 = 0.95$. The blue dotted line indicates the Power function that fits to the plots of the 15 mouse users performance. Figure 6.14 shows the log-log scale of the learning curve.

Table 6.17. The mean task completion time of 15 mouse participants, performing the subtask 1, for learning sessions.

		Day 1	Day 2	Day 3	Day 4
Mouse (M)	*Mean*	25.42	16.49	14.82	13.78
	SD	13.82	8.62	5.98	3.94
	SE	3.57	2.23	1.55	1.02

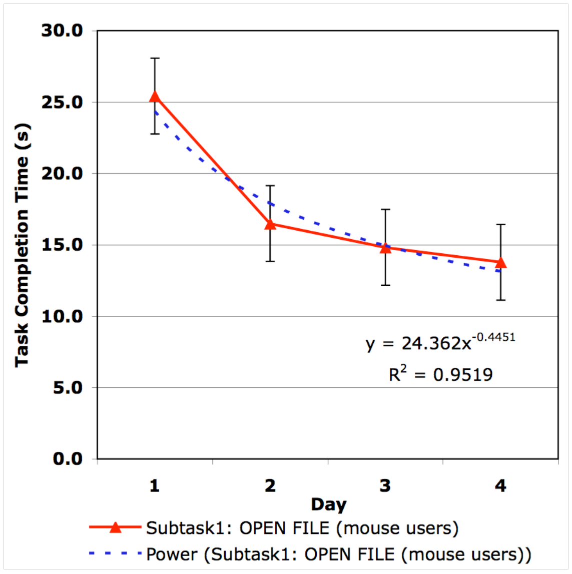


Figure 6.13. The mean task completion time with standard error bars to represent learning performance of mouse users performing the subtask 1 (*n* = 15).

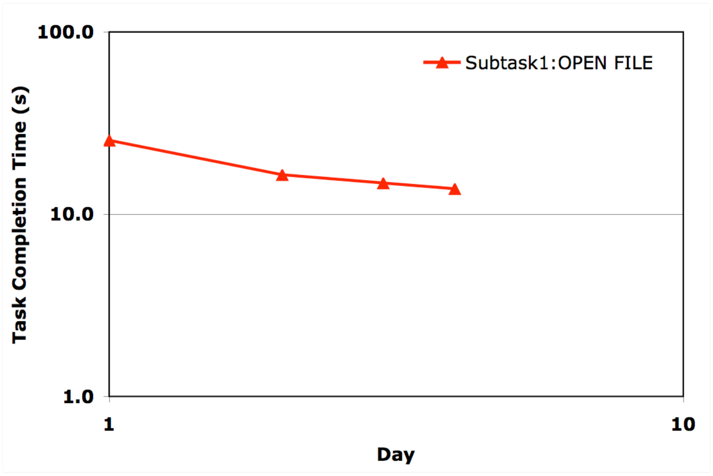


Figure 6.14. The log-log scale of the learning curve.

Figure 6.15 shows plots of learning and forgetting performance by three types of retention intervals (R6, R12, and R18). The performance on the first return day with a six-day retention did not produce forgetting but learning by 1 s decrease in task completion time, that is, 13.5 s (±4.8) on Day 4 and 12.5 s (±2.6) on Day 10. The performance on the first return with a 12-day retention increased by 3 s, that is 14.5 s

(±4.8) on Day 4 and 17.9 s (±8.1) on Day 16. The performance on the first return with a 18-day retention increased by 5 s, that is 13.3 s (±2.7) on Day 4 and 18.4 s (±5.7) on Day 22. Paired samples t-test revealed that there is only significant difference in the task completion time on Day 4 and Day 22, $t(4) = -2.83$, $p < .05$.

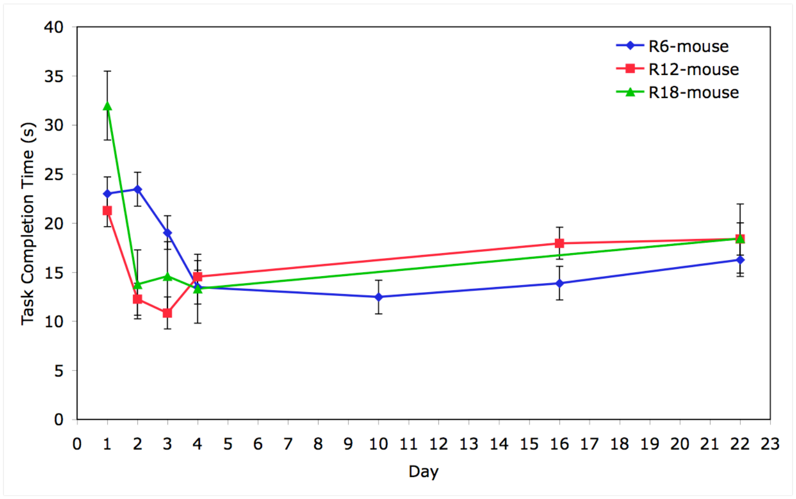


Figure 6.15. Learning and forgetting performance (the mean task completion time with standard error bars) of 15 mouse users, performing the subtask 1.

6.4 Analysis on Speed and Accuracy

I examined whether there is a speed-accuracy tradeoff. To judge the accuracy of the performance, an error-free expert behavior, based on the Keystroke-level model (see Chapter 3), was used as a reference. Table 6.18 shows the number of actions (keystrokes, mouse move, and mouse clicks) from the task analysis that can be used to compare the number of errors.

Mouse Users

I counted the number of mouse clicks that mouse users made during the task performance. Based on the KLM-GOMS analysis, the Dismal spreadsheet task is completed by 125 mouse clicks and 125 mouse movements. When it comes to an accurate performance, the number of mouse clicks that exceeds 125 can be viewed as mistakes. In addition, I also counted the number of "DELETE" keys during the task performance. The number of mouse clicks indicates the accuracy when using the vertical mouse, and the number of "DELETE" keys indicates the accuracy of using the keyboard.

Table 6.18. Summary of the error-free expert behavior.

	Mouse users	Keyboard users
Keystrokes	730	1,030
Mouse move	125	N/A
Menu Selection	87	N/A
Others	38	N/A
Mouse clicks	125	N/A
Menu selection	87	N/A
Others	38	N/A

Table 6.19 shows the number of mouse clicks that the vertical mouse users made during the four days of learning sessions. An expert would make 125 times of mouse clicks. The number of mouse clicks is greater than that of the expert user, indicating more training would be needed for participants to acquire expert performance.

Table 6.19. The mean number of mouse clicks during the four serial learning sessions (*n* = 15).

	Day 1	Day 2	Day 3	Day 4
Mean	172.13	154.40	148.13	153.60
SD	35.86	16.82	20.61	21.65
SE	9.26	4.34	5.32	5.59

Figure 6.16 shows plots of the mean number of mouse clicks. The number of mouse clicks decrease until Day 3, but the number increases on Day 4. As I mentioned in the Section 6.2.1, the learning performance followed the Power law, indicating continual decrease in the task completion time. The mean task completion time on Day 4 by mouse users is 697 s, which is the fastest performance. However, on Day 4, mouse users made more errors in using the vertical mouse.

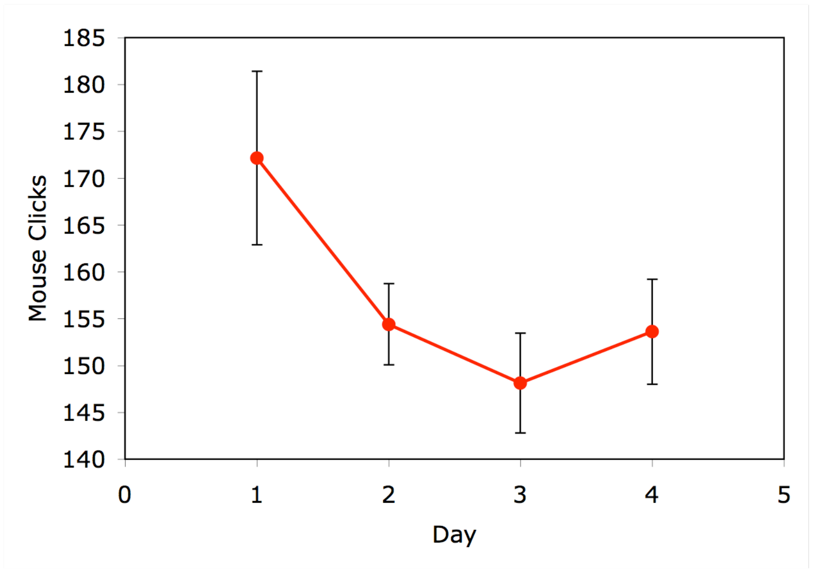


Figure 6.16. The mean number of mouse clicks by fifteen mouse users during the four serial learning sessions with standard error bars.

Table 6.20 and Figure 6.17 show that how many times mouse users made a press on the "DELETE" key. Perfect performance would not press the "DELETE" key, but all participants made a press on that key. During the learning session, the average number of pressing the "DELETE" key decreases except for the number on Day 4. This pattern is similar to the average number of mouse clicks, as seen in Figure 6.16 and Figure 6.17. The correlation between the number of mouse clicks and delete keys is high, $R^2 = 0.98$.

Table 6.20. The mean number of pressing the "DELETE" key during the four serial learning sessions (*n* = 15).

	Day 1	Day 2	Day 3	Day 4
Mean	45.00	33.07	27.73	29.93
SD	28.34	14.33	15.47	18.68
SE	7.32	3.70	4.00	4.82

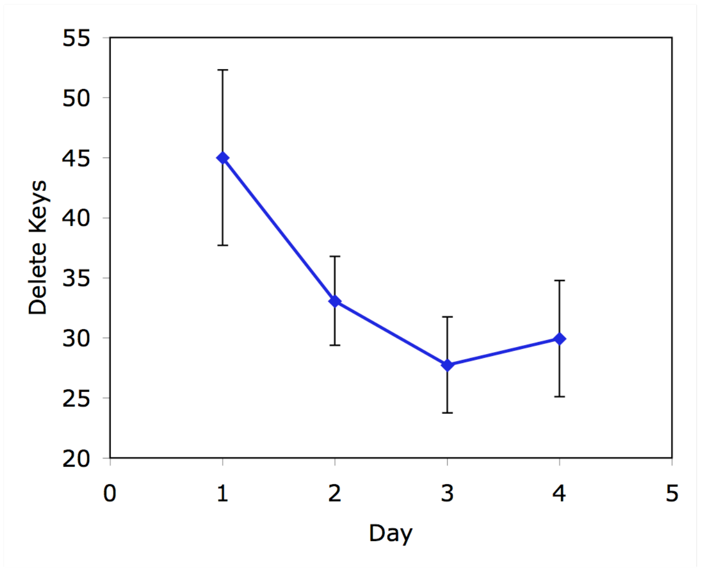


Figure 6.17. The mean number of pressing the "DELETE" key by fifteen mouse users during the four serial learning sessions with standard error bars.

Table 6.21 shows the mean number of mouse clicks made by mouse users for the first return day (e.g., Day 10 for the 6-day retention group, Day 16 for the 12-day retention group, and Day 22 for the 18-day retention group). All of three retention groups showed increase in the number of mouse clicks. Paired-samples t-test revealed that there is only significant difference between the number of mouse clicks on Day 4 and Day 10, $t(4) = -3.1$, $p < .05$.

Table 6.21. The number of mouse clicks on the first return day.

	Retention	Day 10	Day 16	Day 22
Mean	6-day (*n* = 5)	227.80		
SD		71.44		
SE		31.95		
Mean	12-day (*n* = 5)		175.60	
SD			31.29	
SE			13.99	
Mean	18-day (*n* = 5)			174.80
SD				10.52
SE				4.70

Table 6.22 shows the mean number of pressing the "DELETE" key during the task performance on the first return day (6-, 12-, and 18-day) after the learning. Paired samples t-test revealed that there is only significant difference between the number of delete keys between Day 4 and Day 22, $t(4) = -3.9$, $p < .05$.

There is no speed-accuracy tradeoff overall. Both errors and time decrease for mouse users and keyboard users with practice. However, the last day maybe have some tradeoffs.

Table 6.22. The number of delete keys on the first return day by mouse users.

	Retention	Day 10	Day 16	Day 22
Mean	6-day (*n* = 5)	86.00		
SD		48.29		
SE		21.59		
Mean	12-day (*n* = 5)		65.00	
SD			37.76	
SE			16.89	
Mean	18-day (*n* = 5)			93.00
SD				46.47
SE				20.78

Keyboard Users

I referred to pressing the "DELETE" key as making a mistake during the task performance for the keyboard users. Keyboard users would use the DELETE key when they pressed a wrong or unintended key. I counted the number of times that keyboard users pressed the DELETE key, and the mean number of keys is shown in Table 6.23.

Table 6.23. The mean number of delete keys made by users during the performance for four serial learning sessions (*n* = 15).

	Day 1	Day 2	Day 3	Day 4
Mean	39.93	33.27	34.07	35.87
SD	22.03	16.91	27.33	26.58
SE	5.69	4.36	7.06	6.86

Figure 6.18 shows how many times keyboard users pressed "DELETE" keys during the task, for four serial learning sessions. While the task completion time continually decreases (see Section 6.2.1), the number of "DELETE" keys (making errors) slightly increases rather than decreases after the learning session on Day 2.

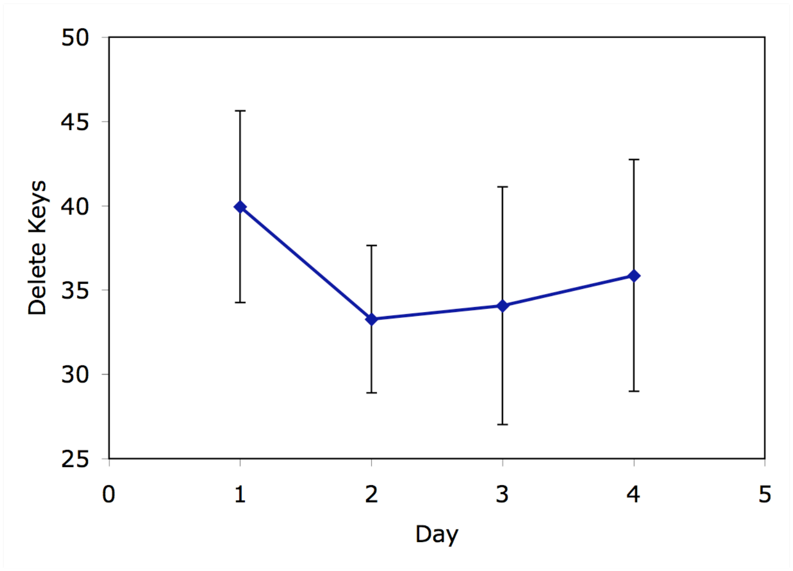


Figure 6.18. The mean number of pressing "DELETE" keys by Keyboard users for four serial learning sessions (*n* = 15) with standard error (*SE*) bars.

Table 6.24 shows the mean number of the "DELETE" key on the first return day. Paired samples t-test revealed that there is only significant difference between the number of the "DELETE" key between Day 4 and Day 22, $t(4) = -4.58$, $p < .05$.

Table 6.24. The mean number of delete keys made by Keyboard users during the performance for four serial learning sessions (*n* = 15).

	Retention	Day 10	Day 16	Day 22
Mean	6-day (*n* = 5)	59.80		
SD		43.96		
SE		19.66		
Mean	12-day (*n* = 5)		71.40	
SD			78.59	
SE			35.15	
Mean	18-day (*n* = 5)			94.40
SD				32.32
SE				14.46

6.5 Understanding the Vertical Mouse

Input modality by using the keyboard and the vertical mouse did not produce any significant differences as seen in the Section 6.2.1 and 6.2.2. This result raises a question that how the vertical mouse is different and novel from the normal mouse.

To address this question, I calculated the index of difficulty (*ID*) of the first subtask for thirteen subjects. In general, the following is offered as the index of difficulty for a motor task:

$$ID = \log_2(2A/W) \qquad \textbf{Equation 6.1}$$

A = the amplitude, which is the distance of the center of the target from the starting location

W = the target width

The index of performance (*IP*) is calculated by dividing *ID* by the movement time (*MT*) to complete a motor task. Also, by regressing *MT* on *ID*, we can get the regression line equation as the following form where *a* and *b* are empirically determined constants, which are obtained by conducting a regression analysis on the movement time data:

$$MT = a + b\log_2(2A/W) \qquad \textbf{Equation 6.2}$$

The *ID* part of Equation 6.2 has variants to improve the data fit to the model, as shown in Equation 6.3 which was proposed by Welford (1968) and Equation 6.4 which was proposed by MacKenzie (1989). In ACT-R, movement time is calculated based on Equation 6.3 without the constant of *a*. MacKenzie (1992) stated that Equation 6.4 prevents *ID* from being a negative.

$$MT = a + b\log_2(A/W + 0.5) = a + bID_1 \qquad \textbf{Equation 6.3}$$

$$MT = a + b\log_2(A/W + 1) = a + bID_2 \qquad \textbf{Equation 6.4}$$

Using the gathered RUI data, I calculated the actual movement time to point and click FILE from home which is the center of the task screen, that is, around x = 840, y = 525 pixels. Figure 6.19 shows the shortest path from home to *FILE*.

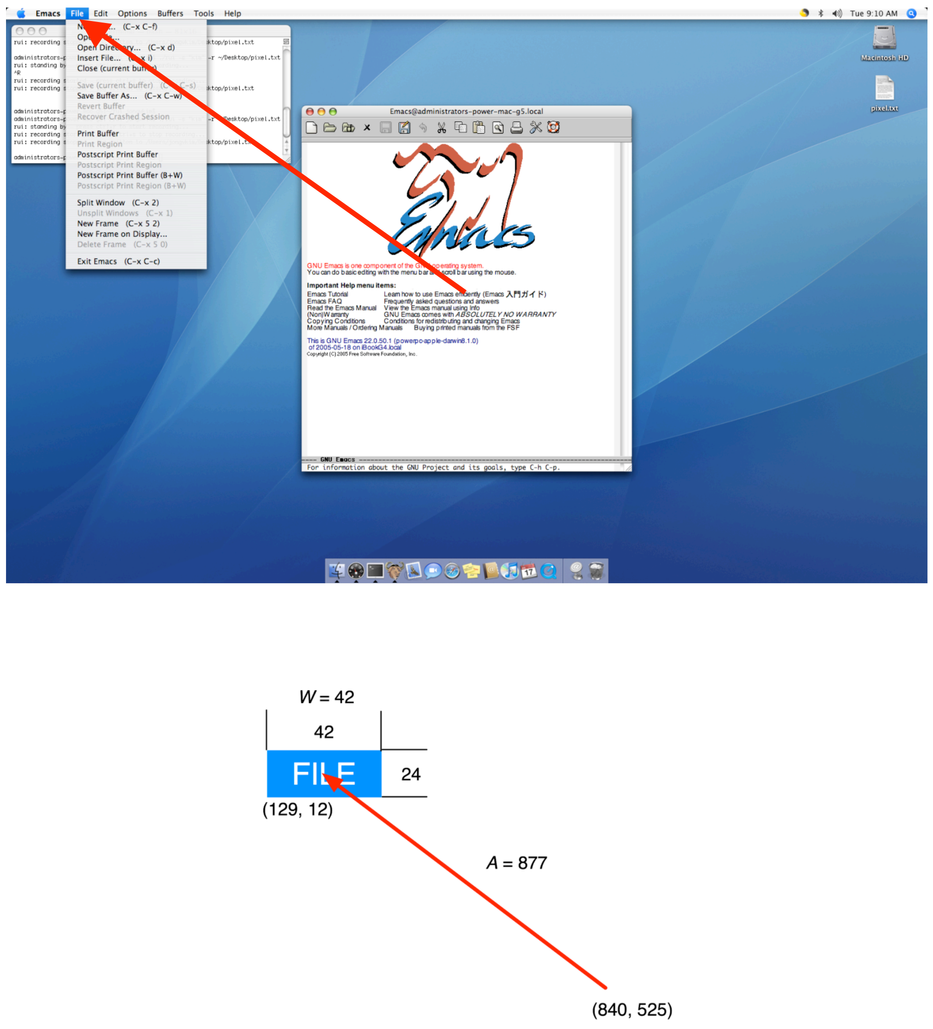


Figure 6.19. The shortest trajectory from home to FILE in the Dismal spreadsheet task.

As seen in Figure 6.19, the amplitude (*A*) is 877 pixels, which is the distance from home to the object (*FILE*) that a user points using the vertical mouse. The target width (*W*) is 42 pixels.

The index of difficulty (*ID*) is calculated based on the three equations (Equation 6.2, 6.3, and 6.4). The *ID* ranges from 4.42 to 5.38. Table 6.25 summarizes different *ID* values. I, in this analysis, chose to use the *ID* equation of $\log_2(A/W+1)$, because it prevents the *ID* being negative.

Table 6.25. Index of Difficulty of pointing the "FILE" object.

Equation	Index of Difficulty (*ID*)
$\log_2(2A/W)$	5.38
$\log_2(A/W+0.5)$	4.42
$\log_2(A/W+1)$	4.45

I found several studies that investigated prediction of the mouse movement time using Fitts' law. In 1978, Card, English, and Burr (1978) evaluated four devices, including a mouse, a rate-controlled isometric joystick, step keys, and text keys, to select text on a CRT display. The distance (*A*) to select the target text from starting point is 1, 2, 4, 8, or 16 cm. The target sizes are 1, 2, 4, or 10 characters[14]. All targets are a group of characters. Card, English, and Burr used a version of Fitts' law by Welford that is Equation 6.3, to find the movement time of different devices. The equation predicting *MT* (in s) for the mouse is $MT = 1.03 + 0.096ID_1$, $R^2 = 0.83$.

MacKenzie, Sellen, and Buxton (1991) also compared input devices (e.g., Macintosh mouse, Wacom tablet and stylus, and Kensington trackball) in pointing and dragging tasks. For pointing tasks, twelve human participants who are computer literate were asked to point an object with $A = 256$ pixels and $W = 16$ pixels. The equation predicting *MT* (in s) for the Macintosh mouse is $MT = 0.107 + 0.223ID$, $R^2 = 0.98$.

Based on the investigation by Card, English, and Burr, when the *ID* is 4.45, the movement time is predicted to be 1.46 s. I can argue that the movement time to point *FILE* using a normal mouse can be 1.46 s. Based on the investigation by MacKenzie, Sellen, and Buxton, when the *ID* is 4.45, the movement time is predicted to be 0.89 s. To compare this movement time with a normal mouse, thirteen participants' learning

[14] One character is 0.246 cm.

performance using the vertical mouse was obtained from the raw RUI data. Table 6.26 shows the mean time for the movement.

Table 6.26. The movement time to point *FILE* using the vertical mouse for four serial learning sessions. (*n* = 13). All units are seconds.

	Day 1	Day 2	Day 3	Day 4
Mean	8.27	4.48	5.83	5.41
SD	5.19	2.77	4.28	2.73
SE	1.44	0.77	1.19	0.76

The movement time of the vertical mouse is larger than that of a normal mouse. Users with a normal mouse could take less than 2 s to this movement, but users with the vertical mouse could take more than 4 s. Longer time to move can indicate harder to use. Thus, it is argued that the vertical mouse that was chosen to use in this dissertation study is different from the normal mouse. The vertical mouse is not only harder to use but also more novel than the normal mouse.

6.6 Summary of Human Data

Participants (*N* = 30) performed a set of spreadsheet tasks. The spreadsheet task was novel enough to measure participants learning and forgetting. All of the participants completed the learning sessions for four serial trials. Both keyboard and mouse users followed and confirmed the Power Law of learning.

Interestingly, both keyboard and mouse users did not lead to significant differences in learning performance over the time range of four days of training and for the population in this dissertation. Similar to the famous log-log linear plot of skill acquisition (Newell & Rosenbloom, 1981), the investigation of the spreadsheet skill acquisition, here, confirmed the Power law of learning.

When it comes to forgetting, the main purpose was to find difference on forgetting performance by two modalities and retention intervals. To investigate the forgetting phenomena, I investigated the first return day after serial trials of learning session. For keyboard users, the 6-day (R6) and 18-day (R18) retention intervals resulted in significant differences on the task performance, *t*(8) = -3.96, *p* < .05. The 12-day (R12) and 18-day (R18) retention intervals had significant differences as well, *t*(8) = 2.29, *p* = 0.051 . For mouse users, the task completion time on R12 vs. R18 produced significant differences, *t*(8) = -3.18, *p* < .05.

Multiple returns for a test after the learning session allowed me to investigate the relearning effect of the Dismal spreadsheet task. There were no significant differences on the learning performance by two modality groups (mouse vs. keyboard users). Unlikely, with regard to relearning, I found interesting results that relearning can be differential between the participants in the mouse group and the ones in the keyboard group. For mouse users, there is a significant difference of the mean task completion time on Day 10 and Day 16, *t*(4) = 3.39, *p* < .05. For keyboard users, there is no significant evidence of the differential relearning effects, *t*(4) = 2.03, *p* > .05. In addition, for mouse users, there is statistical significance of difference the mean task time on Day 16 and Day 22, *t*(4) = 3.32, *p* < .05. For keyboard users, I have to embrace there is no statistical difference by the relearning, *t*(4) = 2.60, *p* > .05. It is found that there is statistical evidence that relearning effects can be affected by the modality and retention interval.

Chapter 7
Comparison of the Model and Human Performance

In this section, I test the ACT-R theory and the Skill Retention Model by comparing the model performance with the human performance.

7.1 Why Do a Comparison?

A model that is based on a sound theory can play a role of a surrogate user in simulating and predicting human behavior. Using a model prediction, if we assume the model prediction is reliable, can be cost effective in decision-making process that involves human operators.

For example, if I want to schedule training sessions for pilots, I need to analyze previous training performance and predict future performance. Because gathering data from pilots is not inexpensive, a model can produce cost-effective data for prediction and scheduling as long as the model provides an accurate representation of human behavior.

Thus, validating a cognitive model is very important issue when it comes to simulation and modeling. A cognitive model validation can provide a better understanding of whether a model has an accurate representation of human behavior.

7.2 How to Validate?

Campbell and Bolton (2005) provided a useful description of how to validate models in two ways: (a) Qualitative validation and (b) Quantitative validation. As qualitative model validation, one can gather validation evidence of the model by asking subject matter experts to assess the model. For example, it is possible to gather validation evidence from a modeling and simulation community (e.g., the ACT-R annual workshop or the ACT-R summer school). This is often referred as *face validity*.

In quantitative model validation, the *goodness-of-fit* measures are used to compare two sets of data. The goodness-of-fit measures can be classified into two types of measures: (a) the trend relative magnitudes, and (b) deviation from exact data location (Schunn & Wallach, 2005). The trend relative magnitude can be described by r^2. Schunn and Wallah (2005) stated that the measures of relative trend are appropriate when the dependent variable is an interval or ratio scale (e.g., frequency counts, reaction time, or proportions). The common ways to capture this trend include Pearson correlation coefficient (r) and r^2. The deviation from data point can be described by *RMSD* (root mean squared deviation) and *RMSSD* (root mean squared scaled deviation). It is important to note that a model can fit the trends of a dataset, but cannot completely capture the exact locations of the data (Schunn & Wallach, 2005).

The most frequently used measure of *the goodness-of-fit* to exact location is the *Mean Squared Deviation* (*MSD*) or the *Root Mean Squared Deviation* (*RMSD*). The model mean is represented by m_i and the data mean is represented by d_i. The number of points i to be compared is represented by k. Other measures to exact location include the *Mean Absolute Deviation* (*MAD*), the *Mean Scaled Absolute Deviation* (*MSAD*), and the *Root Mean Scaled Deviation* (*RMSD*). These measures are listed in Table 7.1.

Table 7.1. Quantitative measures of the goodness-of-fit to exact data location.

Measure	Equation
Mean Squared Deviation (*MSD*)	$\frac{\sum_{i=1}^{k}(m_i - d_i)^2}{k}$
Root Mean Squared Deviation (*RMSD*)	$\sqrt{\frac{\sum_{i=1}^{k}(m_i - d_i)^2}{k}}$
Mean Absolute Deviation (*MAD*)	$\frac{\sum_{i=1}^{k}\lvert m_i - d_i\rvert}{k}$
Mean Scaled Absolute Deviation (*MSAD*)	$\sum_{i=1}^{k}\frac{\lvert m_i - d_i\rvert\sqrt{n_i}}{ks_i}$
Root Mean Squared Scaled Deviation (*RMSSD*)	$\sqrt{\sum_{i=1}^{k}\frac{(m_i - d_i)^2 n_i}{ks_i^2}}$

Note: s_i indicates the standard deviation for each data mean, and n_i indicates the number of data values contributing to each data mean d_i.

For the comparison of the model to data, I am using two measures of the relative trend magnitude (r^2) and the deviation from the exact data location (*MSD*, *RMSD or RMSSD*).

$$MSD = \frac{\sum_{i=1}^{k}(m_i - d_i)^2}{k}$$ **Equation 7.1**

$$RMSD = \sqrt{MSD} = \sqrt{\frac{\sum_{i=1}^{k}(m_i - d_i)^2}{k}}$$ **Equation 7.2**

$$RMSSD = \sqrt{\sum_{i=1}^{k}\frac{(m_i - d_i)^2 n_i}{ks_i^2}}$$ **Equation 7.3**

7.3 Results and Discussion

This section presents results by comparing the human data with the model data. First, I compared the mean task completion time of 15 mouse users and 15 keyboard users performing the Dismal spreadsheet tasks with the learning theory of the Power Law. Second, I compared the 15 mouse users data performing the subtask 1, *OPEN FILE*, with the Skill Retention Model performance.

7.3.1 Power Law of Learning with Data

I tested the theory of Power law of learning, on which ACT-R is basically grounded, against the human data. The data were gathered from mouse users (n = 15) performing the Dismal spreadsheet tasks (all subtasks). Figure 7.1 shows plots of data and the Power curve.

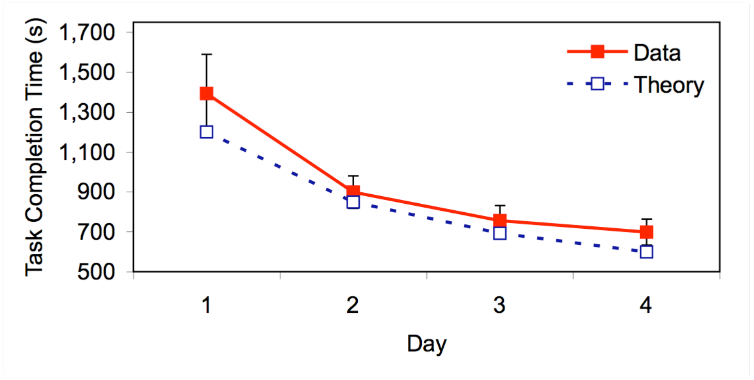


Figure 7.1. Plots to compare the mouse users' data with the Power curve. r^2 is 0.98, *RMSD* is 115.1, and *RMSSD* is 2.2.

The Power curve to fit is $A + BN^{-b}$, with $A = 0$, $B = 1200$, and $b = 0.5$. N indicates the trial numbers in integers. The R^2 is 0.98, indicating a good trend between data and the theory. The *RMSSD* is 2.2, indicating a good location between data and the theory. It is concluded that the Power law can explain mouse users' learning performance for this Dismal spreadsheet task.

Figure 7.2 shows plots of keyboard users' data and the Power curve. The data were gathered from keyboard users (n = 15) performing the Dismal spreadsheet tasks (all subtasks). The Power curve to fit is $A + BN^{-b}$, with $A = 0$, $B = 1500$, and $b = 0.5$. The R^2 is 0.99, showing a good trend between data and the theory. The *RMSSD* is 2.8, indicating a good location between data and the theory. Like mouse users' performance, keyboard users' learning performance is well fitted to the Power law of learning.

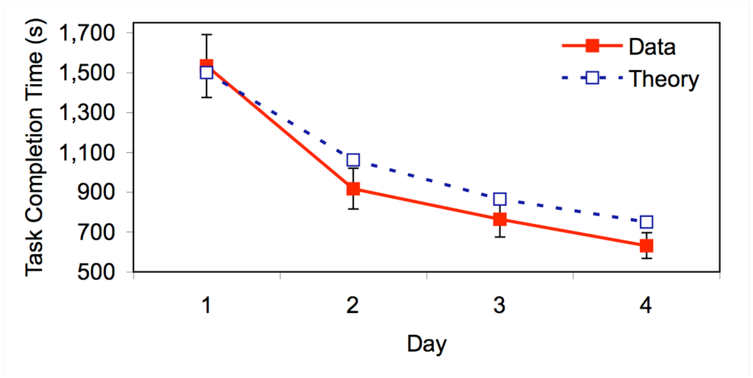


Figure 7.2. Comparison of the keyboard users' data with the Power curve. r^2 is 0.99, *RMSD* is 107.5, and *RMSSD* is 2.8.

7.3.2 The ACT-R Skill Retention Model with Data

I compared the ACT-R Skill Retention Model with human data. The model performance was gathered from the 15 model subjects performing the subtask 1, OPEN FILE, from the Dismal spreadsheet tasks. Human subjects (n = 15) also performed the same task the model did.

Figure 7.3 shows plots of comparing the model with data, resulted in R^2= 0.78, and *RMSSD* = 1.8. The model can predict 78% of the human learning in this specific task. Thus, it is concluded that the ACT-R Skill Retention Model acceptably predicted the learning performance of the subtask 1 in the Dismal spreadsheet task. However, this result leaves challenges that how we can increase the predictability of models for human behavior representations.

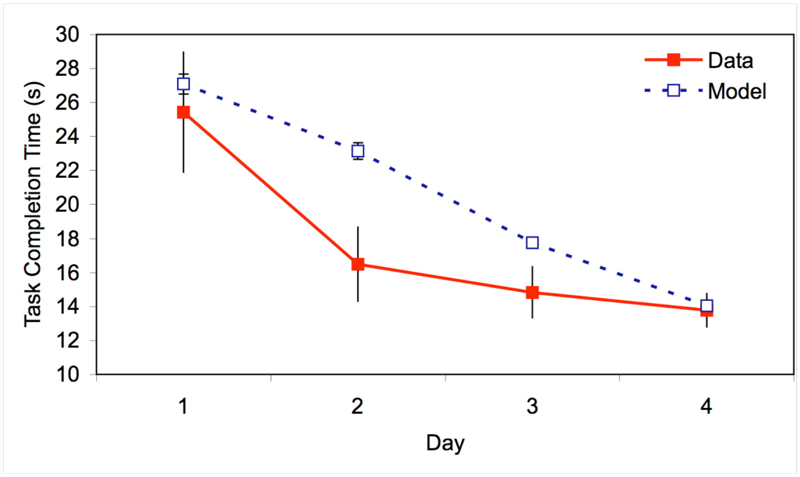


Figure 7.3. Comparison of the model with data.

7.4 Summary of the Comparison

The ACT-R model and its theory were tested against the human data that were gathered and investigated in Chapter 5. When it comes to the learning theory in ACT-R, the Power Law of learning gave a satisfactory comparison (i.e., a good trend and a good location) with the human data of a whole set of the Dismal spreadsheet tasks, $r^2 = 0.98$ and *RMSSD* = 2.2 for mouse users, and $R^2 = 0.99$ and *RMSSD* = 2.8 for keyboard users.

The first subtask of *OPEN FILE* was modeled in ACT-R and the model successfully predicted the learning performance. However, testing forgetting performance of the model is not able to be completed in this dissertation study due to the current scarcity of forgetting mechanism in ACT-R. In the human data, I found that the mean task completion time followed the Power Law as seen in Figure 6.16. The model was tested against the data, r^2=0.78 and *RMSSD* = 1.8. I argue that this prediction for learning performance by the ACT-R skill retention model is quite successful and satisfactory.

Chapter 8
Conclusions

This chapter summarizes findings, contributions, implications, and future work arising from investigating learning and forgetting of knowledge and skills in this dissertation study. As noted earlier in Chapter 1, the motivating problem in my dissertation study is procedural knowledge and skills decay. To better understand learning and forgetting skills, I took two major approaches: (a) Build a cognitive model, and (b) Investigate learning and forgetting skills in a laboratory setting.

Psychological experiments have been a dominant method to investigate and understand human behavior. Also, there has been an active development towards a unified theory of cognition to model human behavior. Both experimental psychology and cognitive modeling have tradeoffs. Psychological experiments in a field can be high-priced and time-consuming. Thus, researchers are forced to emulate a field study inside a laboratory but even this lab setting experiment itself might not provide reliable results because of sample size. To atone for biased experimental data, I used a cognitive modeling to test data and theory.

8.1 The Skill Retention Model in the ACT-R Architecture

In this thesis, I explored the ACT-R architecture and its theories on cognition. The strength of the ACT-R architecture is to provide an ability to model and represent embodied cognition of humans, using various modules and buffers.

This dissertation study found several answers to research questions. I modeled the first subtask (*OPEN FILE*) from the Dismal spreadsheet task. The model predicted learning performance by human participants, $r^2 = 0.8$ and $RMSSD = 1.8$, even though finer tuning of the model is needed for much closer correspondence with human data. The success of modeling the first subtask can guarantee a promising success to represent learning behavior of the real procedural task in a cognitively plausible way.

The skill retention model also highlights some important issues of the ACT-R theory and architecture: (a) limitations of direct connection to the real task environment, and (b) limitations of modeling skill decay.

Modeling of skill decay requires more investigation. When it comes to modeling limitations of skill decay, the investigation in this dissertation confirmed the ultimate need to create a special module or modifications to existing modules in ACT-R to model skill decay. As mentioned in Chapter 3, learned production rules are not able to be unlearned in the ACT-R architecture. However, I explained one way to model decay of procedural knowledge. The current ACT-R architecture supports decay of declarative memory elements by the activation mechanism. Even though the activation mechanism is designed to explain declarative memory elements rather than procedural memory elements (production rules), it is possible to degrade firing of each production rule because procedural memory elements refer to declarative memory elements. That is, the activation of those declarative elements interacts with procedural knowledge. Thus, procedural knowledge can be primed or unprimed by activating or deactivating declarative memory elements.

As mentioned in Chapter 4, cognitive models fail to interact with an external task environment (Ritter, Baxter, Jones, & Young, 2000), although ACT-R/PM (see Byrne, 2001) has helped cognitive models to be equipped with perceptual/motor performance interacting with the environment. To enable cognitive models to perform interactive tasks, it is necessary for the models to have visual perception and motor action capabilities. These capabilities allow a cognitive model to perceive what is on the screen and to make some types of mouse movements.

I attempted to resolve this problem and opened a possibility to help the ACT-R model to interact with a real task environment in the Emacs text editor. The theory of simulated eyes and hands was developed to implement ESEGMAN. All components for the ESEGMAN system were designed. Full implementation of the ESEGMAN system will be done. The ESEGMAN system can provide a continuum of real spreadsheet tasks between the human and the model subjects (Kim, Ritter, & Koubek, 2006). Furthermore, various types of cognitive models in ACT-R can be more effectively embodied to provide a better understanding of user behavior (Kim, Ritter, & Koubek, 2006).

Also, when developing the Skill Retention Model, the Spiral model development process (see Boehm & Hansen, 2001) was used. I extended it by including the factor of technology gaps. Identification of risk factors in three dimensions such as time, cost, and technology gaps helps the risk-driven process of the model development become more reliable.

The strengths of the ACT-R architecture can make contributions to training and education. As I mentioned before, the ACT-R learning mechanisms (e.g., the activation mechanism) help us to better understand prediction of human learning behavior. Based on the modeling and simulation of human performance in ACT-R, scientific management of training programs can be accomplished to shape and steer vital workforce members in military and industry contexts.

8.2 Human Data to Investigate Learning and Forgetting

I created a study environment to investigate learning and forgetting procedural skills in a laboratory setting. The study environment consists of a task in a novel spreadsheet and a tool for measuring human behavior (e.g., mouse clicks, mouse moves, and key presses).

The Dismal spreadsheet task allows us to examine two sets of knowledge and skills, that is, procedural or declarative, and cognitive or perceptual-motor skills (Kim, Koubek, & Ritter, 2007). Furthermore, the task can be modified to support investigations of different types of skills.

I used RUI (Recording User Input) as a tool to timestamp user behavior in an unobtrusive way (Kukreja, Stevenson, & Ritter, 2006). (Kim & Ritter, 2007) reported that using RUI in a study in a naturalistic setting raises new issues. The first issue is that recording can cause a problem when it is used in a public cluster (e.g., a computer classroom). A university policy should and Penn State's policy does prohibit installing any tool for experimentation that obtains a user's identifying information (e.g., a login id or password). Kim and Ritter addressed this problem when RUI is used in public clusters by providing a simple shell script. Using RUI on a jump drive and a shell script programming cuts provides a way to efficiently use RUI on public cluster machines.

The study environment provided reliable testing results of learning and forgetting. The Dismal spreadsheet task was novel enough to measure learning effects from human participants. Analysis on the index of difficulty (see Section 6.5) revealed that the vertical mouse was also novel and different from a normal mouse. The learning performance of participants confirmed that the Power Law of learning applies to this relatively large cognitive task (cf. Newell & Rosenbloom, 1981). There were also no speed-accuracy tradeoffs in learning the first subtask by mouse and keyboard users.

Interestingly, the learning data produced no significant differences on two modalities (mouse and keyboard). During learning the keystroke and mouse driven interfaces were equally easy to learn and equally fast. This is slightly surprising, as many interface designers have argued for the superiority of menu driven interfaces over keyboard driven interfaces (e.g., Shneiderman, 1983).

Three types of retention intervals (R6, R12, and R18) also provided a successful data set to investigate forgetting phenomena of humans. Clear forgetting effects were observed by retention intervals and the forgetting rates are nonlinear.

In addition, relearning effects were investigated in this study environment. It was found that there were significant differences on relearning performance by modalities and retention intervals. Mouse users showed significantly decreased mean task completion time for relearning. This provides an implication that the graphic user interface (GUI) could not provide benefits of easy to learn with less forgetting but provide benefits of relearning.

There was a limitation on the sample size of human participants. I had difficulty in recruiting participants and keeping all of them until they completed the assigned tasks for four weeks. The reason is that participants were asked to make multiple returns (one, two, or three) for measures of forgetting. There remains a need to increase the sample size and a need to add more prolonged retention intervals (e.g., one month or longer).

8.3 Realizing Importance on Training

A movie, *American Fighter Pilot* (Scott, Scott, & Negron, 2002) helped me to understand importance of training. This movie presents that American fighter pilots participated in 110 days of intensive training drills to become a pilot for F-15 Eagle. The training was held at Tyndall Air Force Base in Florida.

According to the movie, it is evident that training fighter pilots requires enormous time, cost, and resources. Fifteen thousand applicants apply to Air Force pilot training school per year. Among them, two thousand are admitted to training. Then, eleven hundred are completed per year. Fifty out of eleven hundred graduates are accepted to Tyndall Air Force Training Base (others go elsewhere). They spend 110 days of F-15 training program. Finally, only eight pilots are qualified as fighter pilots for F-15 and serve the United States.

During the intensive training, the candidates learn detailed information on the F-15 system including hydromechanics, avionics, radar theory, fuel systems, hydraulics, propulsion, instruments, cockpit management, flight controls, navigation, and electronics. The candidates are trained to acquire the systems completely. Acquiring these skills are critical to efficiently respond to emergency situations, such as engine fire during taking-off, under time-critical environments of combat. One little piece of information is important to directly handle aircraft malfunction. Also, seconds are often critical in warfare. After completing the academics of the F-15 system, the candidates practice skills in a simulated cockpit, such as taking-off skills, landing skills, or handling engine fires during taking-off.

The theoretical construct of knowledge and skills degradation can also be applied to the training of these fighter pilots. There might be situations where acquired skills are dormant and susceptible to decay, leading to warfighter performance decrement. This would cause loss of our investment on training.

In this dissertation study, I investigated procedural skills degradation from a theoretical perspective. The power law of learning again confirmed humans' learning behavior. Also, over time, different forgetting rates were observed. Modality difference on forgetting skills should remain tentative because of the small number of participants, but types of skills are intuitively hypothesized to have different forgetting rates.

The infrastructure of the training system can be comprised of manpower, hardware, or software. Optimizing these elements of the training system can reduce any loss on financial investment on manpower, hardware, and software.

Now, let us consider financial aspect on the hardware of the F-15 system. Suppose that the expense of one F-15 aircraft is $500,000,000 and the lifetime of the airframe is 25,000 hours. Thus, with the finite lifetime of the aircraft, the depreciation value is $20,000 per hour.

We can intuitively presume that the cost of training warfighters would be money-demanding. Also, the cost on hardware of the F-15 is expensive. To optimize the cost-benefit tradeoffs, it is necessary to approach the problem from the perspective of multi-disciplines such as cognitive psychology/science and operations research.

8.4 Future Work

My dissertation study touches various important theoretical issues in cognitive modeling and simulation to investigate learning and forgetting of procedural skills. Also, the dissertation study gives us the need for further investigation. Here I specify the list of current and future work to be done.

The skill retention model will be extended to represent all subtasks of the Dismal spreadsheet task. The complete skill retention model will help to understand the general performance of relatively large procedural task and to address each subtask performance.

As noted earlier, ACT-R is limited to declarative memory decay through the activation mechanism. Modeling of skill decay will be accomplished by creating a module or modifying a relevant module based on the ACT-R learning mechanisms (the activation mechanism and the utility learning mechanism). It is expected that the new module is able to cope with degradation of learned procedural memory elements.

In addition, the skill retention model will be connected to the task environment of the Dismal spreadsheet through the ESEGMAN system. Then, a cognitive model with ESEGMAN can have more embodied cognition capabilities and can be a surrogate user representing human behavior.

The sample size of the human data collection ($N = 30$) resulted in failing to reject some hypotheses, providing the need to conduct a larger study to investigate learning and forgetting. For example, I failed to reject difference in learning performance by two modality groups (keyboard and mouse). Also, there was some statistical insignificance observed in forgetting. Thus, increasing the sample size can help to reliably investigate the learning and forgetting. More participants will be recruited to gather more learning and forgetting data.

In this dissertation, I only reported the analysis on the first subtask from mouse users. For the future work, I will analyze all subtasks to obtain much deeper understandings of skill decay. The subtask analyses can provide us with varying attributes of forgetting. It is also necessary to include longer retention intervals (e.g., 30 days or 90 days) in the human study. This helps us to understand and investigate errors that participants will make because of their forgetting. Furthermore, the skill retention model can address errors made by humans.

I will investigate knowledge attributes of various subtasks in the set of tasks here to provide implications on forgetting. For example, there could be differences of learning and forgetting between the subtask of calculations using a normalization equation and opening a file. The former is more a cognition-demanding task than the latter that is simple declarative knowledge retrieval. Deeper understandings on attributes of skills by each subtask can provide important implications on how to plan and design training programs that are relatively resistant to decay. Decay-resistant skills training can augment human performance despite skill disuse over time.

Also, I will need to investigate how the keystroke and mouse move times changed with forgetting. Did, for example, the Fitts' law constant change with forgetting? Did the simple keystroke level times that can be derived from an ACT-R model on this task increase with the forgetting interval?

Analysis on the index of difficulty will be extended to include all subtasks with all participants. In addition, the learning performance of the vertical mouse will be compared to the performance of a normal mouse. There were no speed-accuracy tradeoffs for the learning performance by keyboard and mouse users. This finding will be further confirmed by examining individual keyboard users' performance.

References

Altmann, E. M., & Schunn, C. D. (2002). Integrating decay and interference: A new look at an old interaction. In *Proceedings of the 24th annual meeting of the Cognitive Science Society* (pp. 65-70). Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6*, 451-474.

Anderson, J. R. (1976). *Language, memory, and thought*. Hillside, NJ: Lawrence Erlbaum.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*(4), 369-406.

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

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erbaum.

Anderson, J. R. (1995). *Learning and memory*. New York, NY: Wiley.

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

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review, 111*(4), 1036-1060.

Anderson, J. R., & Bower, G. H. (1973). *Human associative memory*. Washington, DC: Winston & Sons.

Anderson, J. R., Fincham, J. M., & Douglass, S. (1999). Practice and retention: A unifying analysis. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25*(5), 1120-1136.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction, 12*(4), 439-462.

Anderson, J. R., & Reder, L. M. (1999). The fan effect: New results and new theories. *Journal of Experimental Psychology: General, 128*(2), 186-197.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science, 2*(6), 396-408.

Anderson, M. C., & Neely, J. H. (1996). Interference and inhibition in memory retrieval. In E. L. Bjork & R. A. Bjork (Eds.), *Memory* (pp. 237-313). San Diego, CA: Academic Press.

Anderson, R. B., & Tweney, R. D. (1997). Artifactual power curves in forgetting. *Memory and Cognition, 25*(5), 724-730.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 2, pp. 89-195). New York: Academic Press.

Baddeley, A. D. (1996). Exploring the central executive. *The Quarterly Journal of Experimental Psychology, 49A*, 5-28.

Baddeley, A. D. (2001). Is working memory still working? *American Psychologist, 56*(11), 849-864.

Baddeley, A. D., Gathercole, S. E., & Papagno, C. (1998). The phonological loop as a language learning device. *Psychological Review, 105*(1), 158-173.

Baddeley, A. D., & Hitch, G. J. (1974). Working memory. In G. A. Bower (Ed.), *Recent Advances in Learning and Motivation* (Vol. 8, pp. 47-90). New York: Academic Press.

Bahrick, H. P. (1979). Maintenance of knowledge: Questions about memory we forgot to ask. *Journal of Experimental Psychology: General, 108*(3), 296-308.

Bahrick, H. P., Bahrick, L. E., Bahrick, A. S., & Bahrick, P. E. (1993). Maintenance of foreign language vocabulary and the spacing effect. *Psychological Science, 4*(5), 316-321.

Berkun, M. M. (1964). Performance decrement under psychological stress. *Human Factors, 6*, 21-30.

Boehm, B., & Hansen, W. J. (2001). The spiral model as a tool for evolutionary acquisition. *The Journal of Defense Software Engineering*.

Brannon, N. G. (2001). *Knowledge degradation.* Unpublished Doctoral Dissertation, Wright State University.

Brannon, N. G., & Koubek, R. J. (2001). Towards a conceptual model of procedural knowledge degradation. *Theoretical Issues in Ergonomics Science, 2*(4), 317-335.

Brown, J. (1958). Some tests of the decay theory of immediate memory. *Quarterly Journal of Experimental Psychology, 10*, 12-21.

Brownston, L., Farrell, R., Kant, E., & Martin, N. (1985). *Programming expert systems in OPS5: An introduction to rule-based programming*. Reading, MA: Addison-Wesley Publishing Company.

Byrne, M. D. (2001). ACT-R/PM and menu selection: applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41-84.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.), *The Atomic Components of Thought* (pp. 167-200). Mahwah, NJ: Lawrence Erlbaum.

Byrne, M. D., & Gray, W. D. (2003). Returning human factors to an engineering discipline: Expanding the science base through a new generation of quantitative methods - Preface to the special section. *Human Factors, 45*(1), 1-4.

Campbell, G. E., & Bolton, A. E. (2005). HBR validation: Integrating lessons learned from multiple academic disciplines, applied communities, and the AMBR project. In K. A. Gluck & R. W. Pew (Eds.), *Modeling human behavior with integrated cognitive architectures: Comparison, evaluation, and validation* (pp. 365-395). Mahwah, NJ: Lawrence Erlbaum Associates.

Card, S. K., English, W. K., & Burr, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics, 21*(8), 601-613.

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

Carlson, J. G., & Rowe, A. J. (1976). How much does forgetting cost? *Industrial Engineering, 8*(9), 40-47.

Carlson, R. A., & Lundy, D. H. (1992). Consistency and restructuring in learning cognitive procedural sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 18*(1), 127-141.

Chong, R. S. (2004). Architectural explorations for modeling procedural skill decay. In M. Lovett, C. Schunn, C. Lebiere & P. Munro (Eds.), *Proceedings of the Sixth International Conference on Cognitive Modeling*. Mahwah, NJ: Erlbaum.

Cooke, N., Durso, R., & Schvaneveldt, R. (1994). Retention of skilled search after nine years. *Human Factors, 36*(4), 597-605.

Dar-El, E. (2000). *Human learning: From learning curves to learning organizations*. Norwell, MA: Kluwer Academic Publishers.

Duffey, R. B., & Saull, J. W. (2003). Errors in technological systems. *Human Factors and Ergonomics in Manufacturing, 13*(4), 279-291.

Farr, M. J. (1987). *The long-term retention of knowledge and skills: A cognitive and instructional perspectives*. Arlington, VA: Springer.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*(6), 381-391.

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Ed.), *Categories of human learning* (pp. 243-285). New York: Academic Press.

Fu, W.-T., & Anderson, J. R. (2004). Extending the computational abilities of the procedural learning mechanism in ACT-R. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society* (pp. 416-421). Austin, TX: Cognitive Science Society.

Fu, W.-T., & Anderson, J. R. (2006). From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology: General, 135*(2), 184-206.

Glenberg, A. M. (1976). Monotonic and nonmonotonic lag effects in paired-associate and recognition memory paradigms. *Journal of Verbal Learning and Verbal Behavior, 15*, 1-16.

Grant, S. C., & Logan, G. D. (1993). The loss of repetition priming and automaticity over time as a function of degree of initial learning. *Memory and Cognition, 21*, 611-618.

Gray, W. D. (2002). Simulated task environments: The role of high-fidelity simulations, scaled worlds, synthetic environments, and microworlds in basic and applied cognitive research. *Cognitive Science Quarterly, 2*(2), 205-227.

Gray, W. D., & Altmann, E. M. (2001). Cognitive modeling and human-computer interaction. In W. Karwowski (Ed.), *International encyclopedia of ergonomics and human factors* (Vol. 1, pp. 387-391). New York: Taylor & Francis, Ltd.

Greeno, J. G. (1964). Paired-associate learning with massed and distributed repetitions of items. *Journal of Experimental Psychology, 67*(3), 286-295.

Hagman, J. D., & Rose, A. M. (1983). Retention of military tasks: A review. *Human Factors, 25*(2), 199-213.

Johnson, G. (2003, May 13). Mock Explosion Launches Bioterror Drill: Five-Day Exercise to Test Readiness, Determine Strengths and Weaknesses. *The Washington Post.*

Kieras, D. E. (1997). A guide to GOMS model usability evaluating using NGOMSL. In M. G. Helander, T. K. Landauer & P. V. Prabhu (Eds.), *Handbook of human-computer interaction* (2nd ed., pp. 733-766). Amsterdam: North-Holland.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*(4), 391-438.

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

Kim, J. W., Koubek, R. J., & Ritter, F. E. (2007). Investigation of procedural skills degradation from different modalities. In R. L. Lewis, T. A. Polk & J. E. Laird (Eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (pp. 255-260). Oxford, UK: Taylor & Francis/Psychology Press.

Kim, J. W., & Ritter, F. E. (2007). Automatically recording keystrokes in public clusters with RUI: Issues and sample answers. In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th Annual Cognitive Science Society* (p. 1787). Austin, TX: Cognitive Science Society.

Kim, J. W., Ritter, F. E., & Koubek, R. J. (2006). ESEGMAN: A substrate for ACT-R architecture and an Emacs Lisp application. In *Proceedings of the 7th International Conference on Cognitive Modeling* (pp. 375-376). Trieste, Italy.

Konoske, P. J., & Ellis, J. A. (1991). Cognitive factors in learning and retention of procedural tasks. In R. F. Dillon & J. W. Pellegrino (Eds.), *Instruction: Theoretical and applied perspectives* (pp. 47-70). New York: Praeger.

Koubek, R. J., Benysh, S. A. H., & Tang, E. (1997). Learning. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (2nd ed., pp. 130-149). New York: John Wiley & Sons, Inc.

Koubek, R. J., Clarkston, T. P., & Calvez, V. (1994). The training of knowledge structures for manufacturing tasks: An empirical study. *Ergonomics, 37*(4), 765-780.

Koubek, R. J., & Salvendy, G. (1991). Cognitive performance of super-experts on computer program modification tasks. *Ergonomics, 34*(8), 1095-1112.

Koubek, R. J., Salvendy, G., & Noland, S. (1994). The use of protocol analysis for determining ability requirements for personnel selection on a computer-based task. *Ergonomics, 37*(11), 1787-1800.

Koubek, R. J., Salvendy, G., Tang, E., & Brannon, N. G. (1999). Development of a conceptual model for predicting skills needed in the operation of new technologies. *International Journal of Cognitive Ergonomics, 3*(4), 333-350.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI: Recording user input from interfaces under Window and Mac OS X. *Behavior Research Methods, 38*(4), 656-659.

Lee, F. J., & Anderson, J. R. (2001). Does learning a complex task have to be complex? A study in learning decomposition. *Cognitive Psychology, 42*(3), 267-316.

Loftus, E. F., & Loftus, G. R. (1980). On the performance of stored information in the human brain. *American Psychologist, 35*(5), 409-420.

Lovett, M. (1998). Choice. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought* (pp. 255-296). Mahwah, NJ: Lawrence Erlbaum Associates.

Lovett, M. C., Reder, L. M., & Lebiere, C. (1999). Modeling working memory in a unified architecture: An ACT-R perspective. In A. Miyake & P. Shah (Eds.), *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control* (pp. 135-182). New York: Cambridge University.

MacKenzie, I. S. (1989). A note on the information-theoretic basis for Fitts' law. *Journal of Motor Behavior, 21*, 323-330.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction, 7*, 91-139.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems* (pp. 161-166). New York, NY: ACM.

Mazur, J. E., & Hastie, R. (1978). Learning as accumulation: A reexamination of the learning curve. *Psychological Bulletin, 85*(6), 1256-1274.

McKenna, S., & Glendon, A. (1985). Occupational first aid training: Decay in cardiopulmonary resuscitation (CPR) skills. *Journal of Occupational Psychology, 58*, 109-117.

Mertz, J. S. (1997). Using a simulated student for instructional design. *International Journal of Artificial Intelligence in Education, 8*, 116-141.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104*(1), 3-65.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81-97.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior*. New York: Holt, Rinehart & Winston.

Mital, A. (1995). The role of ergonomics in designing for manufacturability and humans in general in advanced manufacturing technology: Preparing the American workforce for global competition beyond the year 2000. *International Journal of Industrial Ergonomics, 15*, 129-135.

Mital, A. (1997). What role for humans in computer integrated manufacturing? *International Journal of Computer Integrated Manufacturing, 10*(1-4), 190-198.

Mital, A., Pennathur, A., Huston, R. L., Thompson, D., Pittman, M., Markle, G., Kaber, D. B., Crumpton, L., Bishu, R. R., Rajurkar, K. P., Rajan, V., Fernandez, J. E., McMulkin, M., Deivanayagam, S., Ray, P. S., & Sule, D. (1999). The need for worker training in advanced manufacturing technology (AMT) environments: A white paper. *International Journal of Industrial Ergonomics, 24*, 173-184.

Myung, I. J., Kim, C., & Pitt, M. A. (2000). Toward an explanation of the power law artifact: Insights from response surface analysis. *Memory & Cognition, 28*(5), 832-840.

Neches, R., Langley, P., & Klahr, D. (1987). Learning, development, and production systems. In D. Klahr, P. Langley & R. Neches (Eds.), *Production System Models of Learning and Development* (pp. 1-53): MIT Press.

Nembhard, D. A., & Osothsilp, N. (2002). Task complexity effects on between-individual learning/forgetting variability. *International Journal of Industrial Ergonomics, 29*, 297-306.

Nembhard, D. A., & Prichanont, K. (2007). Factors affecting cross-training performance in serial production systems. In D. A. Nembhard (Ed.), *Workforce cross training* (pp. 87-109). Boca Raton: FL: CRC Press.

Nembhard, D. A., & Uzumeri, M. V. (2000a). An individual-based description of learning within an organization. *IEEE Transactions on Engineering Management, 47*(3), 370-378.

Nembhard, D. A., & Uzumeri, M. V. (2000b). Experiential learning and forgetting for manual and cognitive tasks. *International Journal of Industrial Ergonomics, 25*, 315-326.

Nevins, J. L., & Whitney, D. E. (1989). *Concurrent Design of Products and Processes: A Strategy for the Next Generation in Manufacturing*. New York: McGraw-Hill.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A. (1993). Desires and Diversions [Video]. Stanford, CA: University Video Communications.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive Skills and Their Acquisition* (pp. 1-55). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

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

Pavlik Jr., P. I. (2005). *The microeconomics of learning: Optimizing paired-associate memory.* Unpublished Dissertation, Carnegie Mellon University, Pittsburgh.

Pavlik Jr., P. I., & Anderson, J. R. (2003). An ACT-R model of the spacing effect. In *Proceedings of the Fifth International Conference on Cognitive Modeling*. Bamberg, Germany.

Pavlik Jr., P. I., & Anderson, J. R. (2004). An ACT-R model of memory applied to finding the optimal schedule of practice. In *Proceedings of Sixth International Conference on Cognitive Modeling*. Pittsburg, PA, USA.

Pavlik Jr., P. I., & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science, 29*, 559-586.

Peterson, L. R., & Peterson, M. J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology, 58*(3), 193-198.

Pew, R. W., & Mavor, A. S. (1998). *Modeling Human and Organizational Behavior: Applications to Military Simulations*. Washington, DC: National Academy Press.

Proctor, R. W., & Dutta, A. (1995). *Skill acquisition and human performance*. Thousand Oaks, CA: SAGE Publications.

Qin, Y., Carter, C. S., Silk, E. M., Stenger, V. A., Fissell, K., Goode, A., & Anderson, J. R. (2004). The change of the brain activation patterns as children learn algebra equation solving. *Proceedings of the National Academy of Sciences, 101*(15), 5686-5691.

Ramos, M. A. G., & Chen, J. J. G. (1994). On the integration of learning and forgetting curves for the control of knowledge and skill acquisition for non-repetitive task training and retraining. *International Journal of Industrial Engineering, 1*(3), 233-242.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering*: North-Holland.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical Conditioning II: Current Research and Theory*. New York: Appleton-Century-Crofts.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., & Schooler, L. J. (2001). The learning curve. In *International Encyclopedia of the Social and Behavioral Sciences* (pp. 8602-8605). Amsterdam: Pergamon.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human and organizational behavior in synthetic environments: A supplemetary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center (HSIAC).

Ritter, F. E., & Wood, A. B. (2005). Dismal: A spreadsheet for sequential data analysis and HCI experimentation. *Behavior Research Methods, 37*(1), 71-81.

Roediger, H. L., & Payne, D. G. (1982). Hypermnesia: The role of repeated testing. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 8*(1), 66-72.

Sabol, M. A., & Wisher, R. A. (2001). Retention and reacquisition of military skills. *Military Operations Research, 6*(1), 59-80.

Schoelles, M. J., & Gray, W. D. (2001). Argus: A suite of tools for research in complex cognition. *Behavior Research Methods, Instruments, & Computers, 33*(2), 130-140.

Schunn, C. D., & Wallach, D. (2005). Evaluating goodness-of-fit in comparison of models to data. In W. Tack (Ed.), *Psychologie der Kognition: Reden and Vorträge anlässlich der Emeritierung von Wener Tack* (pp. 115-154). Saarbrueken, Germany: University of Saarland Press.

Scott, T., Scott, R., & Negron, J. (2002). American Fighter Pilot [DVD]. Fayetteville, AR: Hannover House.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer, 16*(8), 57-69.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University.

St. Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. In *HCI International 2005*. Las Vegas, Nevada.

Stokes, A. F. (1995). Sources of stress-resistant performance in aeronautical decision making: The role of knowledge representation and trait anxiety. In *Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting* (pp. 887-890). Santa Monica, CA: Human Factors and Ergonomics Society.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

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

Taatgen, N. A., Lebiere, C., & Anderson, J. R. (2006). Modeling paradigms in ACT-R. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. New York, NY: Cambridge University.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45*(1), 61-76.

Tarr, R. (1986). Task analysis for training development. In J. A. Ellis (Ed.), *Military contributions to instructional technology*. New York: Praeger.

Welford, A. T. (1968). *Fundamentals of skill*. London: Methuen.

Wisher, R. A., Sabol, M. A., & Ellis, J. A. (1999). *Staying Sharp: Retention of Military Knowledge and Skills* (ARI Special Report 39): The U.S. Army Research Institute for the Behavioral and Social Sciences.

Wisher, R. A., Sabol, M. A., Sukenik, H. K., & Kern, R. P. (1991). *Individual Ready Reserve (IRR) Call-Up: Skill Decay* (Research Report 1595): The U.S. Army Research Institute for the Behavioral and Social Sciences.

Young, R. M., & Lewis, R. L. (1999). The Soar cognitive architecture and human working memory. In A. Miyake & P. Shah (Eds.), *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control* (pp. 224-256). New York: Cambridge University.

Appendix A: How to obtain the software used in this dissertation study

All of the software, presented here for the learning and forgetting investigation, is available without cost. I will briefly provide information on how to obtain the software and to set up the experimental environment. The software is useful for cognitive scientists, human factors engineers, or psychologists to do research or to design a course in academia.

The Lisp implementation that I have used is OpenMCL. OpenMCL is maintained by Clozure Associates (www.clozure.com) and came to have a new name Clozure CL (ccl). My OpenMCL starts up with Emacs with SLIME, providing faster startup speed than other implementations.

Emacs

Emacs is more than a text editor. Indeed, Emacs serves as an operating system in my study environment. You should get the Carbon Emacs Package. I think this is the easiest way to install Emacs in your machine. Please visit here: http://www.apple.com/downloads/macosx/unix_open_source/carbonemacspackage.html

SLIME

SLIME (Superior Lisp Interaction Mode for Emacs) is a mode for Emacs to develop a system using Common Lisp. Explore more information here: http://common-lisp.net/project/slime/

OpenMCL (= Clozure CL = ccl)

I use OpenMCL for my Lisp programming. OpenMCL is an open source Common Lisp implementation. Please visit here: http://trac.clozure.com/openmcl

You can install OpenMCL via MacPorts. MacPorts is an open-source project to provide an easy-to-use system for compiling, installing, and upgrading either command-line, X11, or Aqua based open-source software on the Mac OS X operating system (for more information, please visit www.macports.org).

ACT-R 6

ACT-R is my platform to develop a cognitive model and to test a theory of embodied cognition. ACT-R is one of cognitive architectures. There is a strong and active community to discuss with about ACT-R and its theory. You could check out the ACT-R's official website (http://act-r.psy.cmu.edu). Also, you can download the software from its website.

You may also want to check out the site of the ACT-R FAQ (Frequently Asked Questions), http://ritter.ist.psu.edu/act-r-faq/act-r-faq.html. I now maintaining the ACT-R FAQ website quarterly with Dr. Frank Ritter. You can get practical answers for your questions.

Dismal

Dismal is a spreadsheet that works with Gnu Emacs. In this dissertation study, I used it as a target task containing multiple attributes of knowledge and skills. Dismal are, in essence of its creation, used to align and manipulate sequential data. Please visit http://acs.ist.psu.edu/dismal/dismal.html

RUI

RUI (Recording User Input) is a lightweight and reliable to record user behavior. Recently, ACS lab members are trying to use RUI in a mobile system (e.g., a palm pilot). For more information, please visit, http://acs.ist.psu.edu/projects/RUI/

Appendix B: Glossary

ACT-R	It stands for Adaptive Character of Thought—Rational. ACT-R is a cognitive architecture that is based on a collection of psychological theory for simulating and understanding human cognition.
ACT*	This is a previous version of the ACT-R architecture.
Common Lisp	It is a programming language and was developed to standardize variants of Lisp. Common Lisp provides specifications for the Lisp language.
Dismal	It was named by representing Dis' Mode Ain't Lotus (Dismal). This is a major mode in Gnu-Emacs that implements a spreadsheet. This means you can do spreadsheet works in Emacs.
EASE	It stands for Elements of ACT-R, Soar, and EPIC. EASE combines strengths of ACT-R, Soar, and EPIC into one hybrid integrated architecture.
Emacs	Emacs is an extensible and fully programmable text editor. Emacs was originally implemented in 1976 by the MIT AI lab. The version of Emacs that was used in this dissertation is a part of the GNU project and was implemented in 1984. The Emacs project is supported by Free Software Foundation.
Emacs Lisp	Emacs Lisp is a dialect of the Lisp programming language that is used by Gnu Emacs.
EPIC	It stands for Executive-Process Interactive Control. EPIC is a cognitive architecture to represent human cognition and action.
ESEGMAN	It stands for Emacs SubstratE: Gate toward MAN-made world. ESEGMAN is the system that is designed to connect an ACT-R model to an external environment in Emacs.
HAM	It stands for Human Associative Memory. This is a title of a book by John Anderson and describes a founding theory of human memory.
OpenMCL	OpenMCL is a Lisp implementation. It has a quite long naming history. In 1984, Coral software started developing a Common Lisp implementation for Macintosh, Coral Common Lisp (CCL). Then, it was renamed to Macintosh Allegro Common Lisp (MACL) in 1987. Apple computer took over MACL in 1988 and called it Macintosh Common Lisp (MCL). Digitool took over it again in 1994. In 1998,

	Digitool open-sourced MCL and called OpenMCL. Recently, it is called *CCL* (Clozure CL). Clozure Associates develops and maintains CCL.
Production compilation	It is an operational mechanism that is used in ACT-R. The production compilation mechanism enables production rules to be learned by combining two rules and creating a new rule.
RUI	It stands for Recording User Input. It was implemented in the Applied Cognitive Science lab at Penn State. RUI records users behavior (mouse move, mouse click, and keystrokes) in milliseconds.
Soar	Soar is a cognitive architecture that is used to develop a system producing intelligent behavior.

VITA
Jong W. Kim

316E IST Building
College of Information Sciences and Technology
Pennsylvania State University, University Park, PA 16802
814.865.6166 juk166@psu.edu

Research Interest

I am interested in investigating research issues related to human factors and cognitive science. Particularly, I have been engaged in simulation and modeling of human performance with respect to knowledge/skills acquisition and degradation. I am using the ACT-R cognitive architecture to model human performance and test the theory by comparing the model with the human performance. I am also interested in developing a novel paradigm of training principles to optimize resources in military and industry.

Education

Ph.D. The Pennsylvania State University, 2008 (Industrial Engineering)
Graduate Coursework, University of Central Florida, 2002 (Industrial Engineering)
M.S. University of Central Florida, 2001 (Industrial Engineering)
B.S. Hong-Ik University, Seoul, Korea, 1999 (Industrial Engineering)

Professional Experience

Post-doctoral Research Fellow, May 08 ~ Current
Project: Investigation of Procedural Knowledge and Skills Degradation, The Office of Naval Research
Named Researcher, June 07 ~ May 08
Establishing a novel training paradigm by investigating learning and forgetting of procedural skills, supervised by Ritter (PI) and Koubek (Co-PI), The Office of Naval Research
Research Assistant
- College of IST, Penn State Univ., supervised by Dr. Frank Ritter
  Publish and maintain Soar FAQ, Jan. 05 ~ May 08
  (acs.ist.psu.edu/projects/soar-faq/soar-faq.html)
- Dept. of Industrial Engineering, Penn State Univ., supervised by Dr. Rick Koubek, Jan 03 ~ Aug. 04
- Dept. of Industrial Engineering and Management Systems, Univ. of Central Florida, supervised by Dr. Gene Lee, Aug. 01 ~ Dec. 02
  Project Title: Enhanced biological/chemical isolation suit with internal cooling systems, Sponsor: Department of Defense

Teaching Assistant
- Dept of IEMS, Univ. of Central Florida, Jan 01 ~ May 01

Attended at the 12th ACT-R Summer School, Carnegie Mellon University, July 2005