Developing Effective Rule-Based Training Using the Cognitive Work Analysis Framework

by

Thomas Alan Robinson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2013

Author's Declaration

Abstract

Cognitive Work Analysis (CWA) is a framework for the analysis of complex systems that involve technological tools and human operators who must operate the systems in sometimes-unexpected situations. CWA consists of five phases that cover analysis of ecological aspects of systems, such as the tools in the system and the environment, and cognitive aspects, related to the human operators in the systems. Human operators often need to be trained to use complex systems, and training research has been conducted related to the ecological aspects of CWA; however, there is a gap in the training research related to the cognitive aspects of CWA. The Skills-Rules-Models (SRM) framework is currently the main method of cognitive analysis for CWA; therefore, to begin to fill the gap in the training literature at the cognitive end of CWA, this dissertation examines the need for training as related to SRM.

In this dissertation, the current research related to training and CWA is reviewed and literature on the nature of expertise as related to SRM is examined. From this review, the need for training rules that can be used in unexpected situations, as a means of reducing cognitive demand, is identified. In order to assist operators with developing the knowledge to use such rule-based behaviour, training needs and methods to meet those needs must be identified.

The reuse of knowledge in new situations is the essence of training transfer, so ideas from training transfer are used to guide the development of three guidelines for determining rules that might be transferable to new situations. Then methods of developing rules that fit those three guidelines by using information from the Work Domain Analysis (WDA) and Control Task Analysis (CTA) phases of CWA are presented.

In addition to methods for identifying training needs, methods for meeting those training needs are required. The review of training transfer also identified the need to use contextual examples in training while still avoiding examples that place too high of a cognitive demand on the operator, possibly reducing learning. Therefore, as a basis for designing training material that imposes a reduced cognitive demand, Cognitive Load Theory (CLT) is reviewed, and methods for reducing cognitive demand are discussed.

To demonstrate the methods of identifying and meeting training needs, two examples are presented. First, two sets of training rules are created based on the results of the application of the WDA and CTA phases of CWA for two different work domains: Computer Algebra Systems and the programming language Logo. Then, from these sets of rules, instructional materials are developed using methods based on CLT to manage the cognitive demands of the instructions. Finally, two experiments are presented that test how well operators learn from the instructional materials. The experiments provide support for the effectiveness of applying CLT to the design of instructional materials based on the sets of rules developed. This combined work represents a new framework for identifying and meeting training needs related to the cognitive aspects of CWA.

Acknowledgements

I wish to thank Dr. Catherine Burns for the invaluable support and guidance that she has provided over the last four years.

I also thank the other members of the Human Factors group for their advice and support. I also thank them for the many fun times we had during sushi trips, camping trips, and HFES conference travels.

Thank you also to the other members of my thesis committee for their advice, comments, and support: Drs. Stacey Scott, Carolyn MacGregor, Jose Arocha, and Ellen Bass.

I also thank my colleagues and managers at Maplesoft for their support during and before this degree work. Thank you also to Maplesoft and NSERC for their financial support of my research.

Finally, thank you to my family for their support and care over many years.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

4C/ID	Four-Component Instructional Design
CLT	Cognitive Load Theory
CTA	Control Task Analysis
CWA	Cognitive Work Analysis
EID	Ecological Interface Design
GUI	Graphical User Interface
JIT	Just-in-Time
SA	Strategies Analysis
SOCA	Social and Cooperation Analysis
SRM	Skills-Rules-Models
WCA	Worker Competency Analysis
WDA	Work Domain Analysis

Chapter 1
Introduction

Sociotechnical systems involve both human and technical components that work together to accomplish work domain goals (Vicente, 1999). Training is an important consideration in sociotechnical systems (Naikar, 2009) and will involve ecological considerations, related to the tools and environment, and cognitive considerations, related to the human operators. During training, operators will need to learn to deal with routine operations, but, even more importantly, they need to deal with disturbances, errors, and unexpected situations (Vicente, 1999; Naikar, 2009). Despite the need for training methods for sociotechnical systems, there is a limited amount of research that combines learning science literature with systems training literature (Fowlkes, Neville, Owens, & Hafich, 2009), albeit some crossover does exist (e.g. Kalyuga, Chandler, & Sweller, 1998; Wickens, Hutchins, Carolan, & Cumming, 2013). Furthermore, Fowlkes et al. (2009) claimed there is a need to increase communication between the learning science and systems training communities so that training needs and solutions can be shared between the communities.

Cognitive Work Analysis (CWA) (Rasmussen, 1986; Vicente, 1999) is one framework used to analyse the requirements for sociotechnical systems. CWA has been used in a number of applications. One of the primary applications of CWA is Ecological Interface Design (EID) (Rasmussen & Vicente, 1989; Vicente, 2002; Burns and Hajdukiewicz, 2004). Recently, CWA has also been used in identifying training needs, comparing competing systems, and organizing teams (Naikar, 2006b, 2009).

CWA consists of five phases: Work Domain Analysis (WDA), Control Task Analysis (CTA), Strategy Analysis (SA), Social Organization and Cooperation Analysis (SOCA), and Worker Competency Analysis (WCA). Each successive phase in a CWA examines how the requirements generated during the previous phase can be fulfilled. During each phase ecological and cognitive factors in the system are considered, but as the phases progress, the focus shifts from ecological to cognitive. The final phase in CWA, WCA, the phase considered most frequently in this dissertation, examines what cognitive abilities are needed by operators in order to complete tasks successfully within a system.

The current literature on the WCA phase relies on Rasmussen's (1983, 1986) Skills-Rules-Models (SRM) taxonomy (Vicente, 1999), which categorizes qualitatively different types of behaviour into skill-, rule-, and model- based behaviour. (The SRM taxonomy is usually referred to as the Skills-

Rules-Knowledge taxonomy; however, for reasons of avoiding confusion, which will be discussed later, the term model is used instead.) The SRM taxonomy can be of use in identifying the cognitive abilities of operators and identifying skill gaps that need to be filled to allow those operators to meet requirements derived from the previous CWA stages (McIlroy & Stanton, 2011). The use of the SRM taxonomy is not limited to CWA and has also been used independent of CWA in medical applications (Dankelman, Wentick, Grimbergen, Strassen, & Reekers, 2004; Wentick, Stassen, Alwayn, Hosman, & Stassen, 2003) and error analysis (Reason, 1990; Naikar and Saunders, 2003). In this dissertation, training methods related to SRM are developed in order to fill a need for research into training at the cognitive end of CWA.

1.1 Problem statement

The above applications of CWA and SRM all involve a component of training. For example, Vicente (2002) noted that one goal of EID is to encourage more rule-based behaviour. Also, in the training papers of Dankelman et al. (2004) and Wentick et al. (2003), methods for training behaviours are categorized into the three SRM behaviour levels. Finally, after completing an error analysis using CWA methods, Naikar and Saunders (2003) suggested that training could be desirable to reduce future error.

Connecting CWA to cognitive training requirements is important because CWA is an approach for understanding cognitive work in complex systems (Vicente, 1999), where operator training is regularly required (Woods & Roth, 1988). CWA is already used in understanding some training requirements related to the ecological components of CWA. For example, Naikar and Saunders (2003) and Naikar (2006b) examined training from a CWA perspective. McIlroy and Stanton (2012) noted the need for training in the context of CWA, and discussed Naikar's (2006b) work. However, these pieces of research did not discuss training in terms of WCA or SRM. Lintern and Naikar (1998) discussed SRM in the context of a CTA for the purposes of training, but they did not discuss methods of identifying training needs in terms of SRM.

The above work considers training requirements from the ecological perspective of CWA; however, to consider the relationship between CWA and training in a complete way, training as related to the WCA phase must be considered because that phase is used in the analysis of the cognitive abilities of the operators who monitor and control the system. Considering training in the context of the WCA phase requires the connection of training methods with SRM, because SRM is the main tool for WCA. Despite the usefulness of relating learning to WCA and the SRM taxonomy, there is work to be done on the connection between the taxonomy and learning (Dankelman et al, 2004).

1.1.1 Outcomes

In this dissertation, to fill in the above-mentioned gap related to cognitive training requirements in CWA, I look at training needs that can be described by SRM and methods of fulfilling those training needs. By using guidance from the learning science literature, in particular training transfer, I examine how training needs can be described by the rule-based behaviours of the SRM taxonomy. In particular, these training needs relate to preparing operators to use rule-based behaviours as part of model-based behaviours in unexpected situations. This work results in three guidelines for identifying rule-based behaviours that could be useful in unexpected situations. The earlier phases of CWA are then used to develop examples of these rules in two example domains, helping to connect the work in this dissertation to the other phases of CWA. In the empirical work in this dissertation, I also examine how Cognitive Load Theory (CLT) can provide guidance in designing training materials to meet the identified training needs while avoiding cognitive overload, which is a risk when training for transfer. As part of the empirical work, two experiments are presented that test the use of CLT-based instructional design in the training of rules derived using the guidelines presented.

The work in this dissertation starts to fill in the gap in the literature for CWA concerning training needs at the cognitive end of CWA by providing a method of identifying rule-based training needs and methods for meeting those needs. Additionally, this work continues to bridge the gap between learning science and systems training research, by providing an example of how concepts from these two communities can be integrated.

1.2 Research Approach

In this section, the approach taken for the research in this dissertation is discussed. First, however, to frame the approach taken, the methods used in human factors research related to CWA are outlined.

1.2.1 Complex Systems and Human Factors

Most modern work environments are complex systems involving a number of components, both human and technological, which work in coordination to accomplish heavily cognitive tasks (Vicente, 1999; Rasmussen, 1986; Hutchins, 1995; Hollan, Hutchins, Kirsh, 2000; Woods, 1988). These systems usually consist of human components, operators, and technological tools that the operators can use to help complete tasks in the system (Vicente, 1999). As Woods (1988) pointed out, important characteristics of complex systems include their dynamic nature, uncertainty, and the interconnectedness of human operators and technology.

Complex systems can be dynamic, meaning that state changes depend not only on operator actions but also on time-dependent occurrences in the environment (Woods, 1988). Because of the range of possible sources of effects on a system, it is difficult to know what components in a system and its environment will be important in a study of a system and, therefore, it is often difficult to know what should be considered internal to a system and what should be considered external. Consequently, studies of systems can begin with the difficult question of system scope (Wood and Roth, 1988; Marmaras & Nathanael, 2005; Proctor and Van Zandt, 2008).

Related to the issue of the dynamic nature of a complex system is the potential for scenarios that involve uncertainty, unknown information, and even wrong information (Woods, 1988; Woods and Roth, 1988). Furthermore, uncertainty can arise from unexpected events ranging from minor disturbances to major crises (Vicente, 1999). Therefore, because of the possibilities for uncertainty, before completing a task in a system, an operator must first determine what task actually needs to be completed. Despite these difficulties in operating a system, the human operators must still determine methods of using the tools in a system to complete tasks (Woods, 1988; Woods and Roth, 1988). In completing such tasks, operators might use tools in unexpected and novel ways (Vicente, 1999; Naikar, Pearce, Drumm, & Sanderson, 2003; Naikar, 2006).

Because systems envisioned by Woods (1988) are collections of human operators and technological components, to complete tasks efficiently, all the components in a system must work together in a highly interconnected, manner (Woods, 1988). Traditionally, theories of cognition have focused on cognition as an aspect of an individual (Hollan et al., 2000), but with complex systems involving high interactivity of operators and tools, Hutchins (1995), Hollan et al. (2000), and Lintern (2009) have argued that the consideration of cognition must be expanded beyond the individual to include the cognition embedded in tools, social structures, and cultural concepts that a person or group of people use in completing a cognitive task. Hutchins (1995) argued that cognition is not just an internal process, but a process of coordination, possibly among several individuals and tools, to complete a task. Lintern (2009) further argued that cognition is really a process of "reciprocity between an organism and its environment" (p. 329). These claims are not saying that cognition of the individual is not important; each human in a system has particular characteristics that suit him or her to completing a particular part of a cognitive task, but the way in which the components of the system coordinate determine what cognitive tasks can be completed in a system (Hutchins, 1995; Lintern, 2009). For example, a system composed of a human and a calculator is much more capable of mathematical problem solving than either the human or the calculator alone (the calculator sitting by itself, unused, is quite incapable of anything, and a human alone would expend much more effort on routine calculations). Therefore, the analysis of cognition in complex systems involves both individual cognitive considerations, related to abilities of individual humans, and ecological considerations, related to the environment of the system, tools in the system, and the interactions between all those components.

As a result of all these factors, studies of systems using human factors methods draw on multiple domains of knowledge and methods (Woods and Roth, 1988; Marmaras & Nathanael, 2005; Proctor and Van Zandt, 2008). This multi-domain approach leads to the International Ergonomics Association and Human Factors and Ergonomics Society's definition of human factors as: *"the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data, and other methods to design in order to optimize human well-being and overall system performance."* (International Ergonomics Association, 2010).

In this dissertation, the focus is on the knowledge operators need to use tools and how they learn that knowledge. The research is, therefore, focused on the third property of Woods' (1988) complex systems definition: the interaction between operators and tools. Because the hope is that this work can be expanded to more complex systems, the research in this dissertation is conducted using human factors research methods that are used for complex systems.

1.2.2 Human Factors Studies and Validity

The complexities involved in the study of complex systems result in the use of a variety of research methods including descriptive studies (such as ethnographic studies eg. Hutchins, 1995; Woods, 1998), error analysis (eg. Reason, 1990), formal theory, and experiments (Proctor & Van Zandt, 2008). While these methods are common in a number of areas of research, Proctor & Van Zandt (2008) point out some distinctions between human factors methods and traditional lab experiments. Human factors research is targeted to studying the reality of work in complex systems, or producing what Proctor & Van Zandt (2008) call "ecologically valid" results, which closely describe real-world situations. Because of the number of factors that can affect a system, it is difficult to control all the possible threats to internal validity and construct validity and still conduct a study that captures the nature of the system (Marmaras & Nathanael, 2005). Therefore, in the attempt to generate ecologically valid results, when a number of complexities can affect results, other types of validity such as internal validity and construct validity (Proctor & Van Zandt; Trochim & Donnelly, 2008) are often sacrificed to some degree in human factors experiments when compared to traditional laboratory experiments (Proctor & Van Zandt, 2008). However, there is still obviously a need to provide some control in any study, to ensure that there are some valid and reliable conclusions that can be drawn, so there is always some trade-off between the ecological validity of a study and study control (Marmaras & Nathanael, 2005; Proctor & Van Zandt, 2008).

In many studies in Human Factors, in particular in studies related to EID (a key application of Skills-Rules-Models taxonomy discussed in this dissertation), there are questions of internal and construct validity. For example, Christoffersen, Hunter, & Vicente (1996) and Vicente, Christoffersen, & Pereklita (1995) both note issues of construct validity related to the operationalization of EID theory in the design of their study interfaces. They note that it is possible that the performance differences measured could be due to changes in display form rather than due to the application of EID principles. This validity difficulty was reiterated by Vicente (2002). Christoffersen, Hunter, & Vicente (1996) and Vicente, Christoffersen, & Pereklita (1995), and other authors, instead support their conclusions using the results of previous work that had investigated the usefulness of EID designs beyond simple changes in display form, a result again summarized in Vicente (2000). Similarly, Burns et al. (2008) use measures of situational awareness (Endsley, Bolté, & Jones, 2003) as a measure of the effect of an ecological interface, though they admit that there are warnings about “blindly transporting” (p. 678) measures of situational awareness to new domains. This last example represents a threat to the construct validity of their measure of situational awareness.

The studies above, and other summarized in Vicente (2002), are examples of studies where existing theories are applied to the design of interfaces for specific problems. One goal of the studies is to demonstrate the application of EID, while at the same time providing some evidence for the general usefulness of EID. This dual purpose illustrates another claim of Woods and Roth (1988) about another feature of human factors studies: domain specificity.

With human factors studies focused on real-world systems, the studies tend to be domain specific (Woods and Roth, 1988). As such, the goals of human factors studies include understanding a work domain, understanding how people solve problems in the work domain, and providing designs to improve performance (including boosting efficiency and reducing errors) (Woods & Roth, 1988; also Long, 2000). The results of domain-specific studies lack generalizability (McGrath, 1984), but another goal of human factors research is the development of general theory and principles that are generalizable to other situations or work domains (Proctor & Van Zandt, 2008; Woods & Roth, 1988; McGrath, 1984). To accomplish this latter goal and support more general conclusions, multiple results of a number of studies can be combined through methods such as triangulation, where multiple related and consistent results are used in support of a more general theory (McGrath, 1984; Creswell & Clark, 2007).

The above shows that there is a trade-off made in human factors studies between ecological validity and other forms of validity including construct, internal, and external validity. Such a trade-off also affects how the results of experiments can be used in the support of more general theories. Therefore, to build a set of results, the work in this dissertation relies on a number of methods, and background research, that are used to support the theoretical work presented and the conclusions drawn.

1.2.3 The Role of Theory in Human Factors Studies

While general theories are a goal of human factors research, they can also guide research (Marmaras & Nathanael, 2005). When used to guide research and the investigation of a system, theories focus attention on certain aspects of the system, but can also blind a researcher to other aspects (Marmaras & Nathanael, 2005). In discussing this trade off, Marmaras and Nathanael (2005) describe two approaches to human factors research: the empirical-positivist and empirical-hermeneutic approaches.

In the empirical-positivist approach, formal theories, models, and methods are used to interpret the real-world system being studied in terms of a study-world (Marmaras & Nathanael, 2005). The theories used help determine the system boundaries, the nature of the work domain, the components of interest in the system, and the goals of the system. These aspects of the system are mapped onto and studied through the lens of the various theories, models, and methods chosen. The empirical-positivist approach can help focus a study on certain aspects of interest of a system, which is an important goal when the alternative of considering all aspects of a system might be overwhelming.

However, with this approach, there is a risk that important aspects might be missed if they are outside the scope of the theories, models, and methods used (Marmaras & Nathanael, 2005).

To avoid inappropriate framing of a study and missing important aspects of a system, a researcher can use an empirical-hermeneutic approach (Marmaras & Nathanael, 2005). In this approach, the study begins with direct consideration of the real-world system without the filter of particular theories or models. The researcher begins the study by building an account of what is happening in the system and trying to recognize patterns in behaviours, through methods such as field studies (McGrath, 1984; Hutchins, 1999; Woods, 1993; Mumaw, Roth, Vicente, and Burns, 2000), as are discussed later in this dissertation in the introduction to a field study. Marmaras and Nathanael (2005) noted that it is important to recognize that behaviours might show up in unexpected places. They gave an example of an informal 11am coffee break that is actually an important moment for coordination of some activity. In order to uncover these, possibly disguised, important aspects in a system, it is necessary to try to see the world from the perspective of an operator in a system and try to uncover the meaning or purpose of their activities. The focus of the empirical-hermeneutic approach is on discovering and representing the functioning of a system. Two drawbacks of the empirical-hermeneutic approach are an overwhelming amount of information that could be generated and the time needed to do a thorough analysis (Marmaras & Nathanael, 2005).

In reality, most analyses of a system will involve a combination of the empirical-positivist and empirical-hermeneutic approaches (Marmaras & Nathanael, 2005). Normally, a researcher will be guided by theory, but will attempt to not be blinded by the theory. Moreover, many analyses will be cyclical, with some observation followed by some reinterpretation of results in terms of appropriate theories and models (Marmaras & Nathanael, 2005).

These various human factors research methods are combined in this dissertation to achieve a number of theoretical and empirical outcomes.

1.2.4 Background Work

After this chapter, a review of literature is presented, including human factors research, mostly focused on CWA and training as related to CWA. Gaps in the literature related to training at the cognitive end of CWA are identified, particularly as related to the SRM taxonomy.

To start to fill in some of these gaps, transfer of training literature is reviewed, and connections between that literature and SRM are made to develop methods of identifying training needs. Finally, CLT literature, which is later used to develop methods of meeting those training needs without imposing too high of a cognitive demand, is examined.

1.2.5 Field Study and Experiment

Before presenting the theoretical work in this dissertation, Computer Algebra Systems, which will serve as an example domain for the theoretical work, are introduced. Computer Algebra Systems are introduced and a note is made that they can impose difficulties on students if not introduced in a well-considered manner. A field study is then presented that investigates some of the specific difficulties that students can experience when using the Computer Algebra System Maple.

Next, a field experiment is presented that compared how experienced Maple users solved mathematical problems using either Maple or pen and paper. The results from this study are used in the design of the experiment, discussed below, to develop a WDA for Maple.

1.2.6 Theoretical Work

The theoretical work in this dissertation is intended to address the gap in the CWA training literature at the cognitive end of CWA. Through the use of training transfer literature, I consider types of behaviours described by SRM that could be useful in future, unexpected situations. The theoretical work results in three guidelines for determining rule-based behaviours that might be candidates for transfer to unexpected situations.

As with any CWA analysis, the results of a CWA are not an end product; the CWA results are often used in a creative design process (Vicente, 2002). The use of the theoretical concepts in this dissertation result in a set of teachable, and, hopefully, transferable, rules, that can then be used in the design of instructional material. Turning the rules into instructional material is a creative design process that involves information from the CWA and the domain to which the CWA is applied. The application of the theoretical work presented in this dissertation is demonstrated through a design example using the results from the Maple field experiment. Through this example, and another example in the empirical work discussed below, the use of the theoretical work is demonstrated and the theory is related back to the first two phases of CWA: WDA and CTA.

1.2.7 Empirical Experiments

Two empirical studies have already been discussed: the field study to assess learning difficulties and a field experiment to investigate experienced user behaviour. Two experiments are also presented that test the training of concepts developed using the theoretical work. The experiments involved instructional design for teaching students to use two different, but related, pieces of technology: Maple and Logo. For both experiments, a set of rule-based behaviours was developed based on the theoretical work presented and a combination of WDA and CTA results. Then, for each experiment, two sets of instructional materials were developed based on CLT. After the participants learned some of the respective technology, they were tested on their learning and their workload during learning and testing. The measures used during testing in the first experiment focused on the evaluation of rule learning, while in the second study the measures focused the participants' ability to apply the knowledge they learned during instruction to new problems presented during testing. These experiments present design examples of the concepts resulting from the theoretical work and test the use of CLT principles in developing the corresponding instructional materials.

1.3 Dissertation Structure

Chapter 1, the current chapter, presents the problem studied in this dissertation and outlines the research approach taken.

Chapter 2 reviews the literature related to CWA, SRM, training transfer, and CLT.

Chapter 3 presents the field study and field experiment. Also in chapter 3, the field study results are used to develop a WDA and CTA for using Maple.

Chapter 4 presents the theoretical work in this dissertation and a design example that uses that theory. The work in chapter 4 results in some general guidelines for the determination of training rules and a set of example rules for the use of Maple that are based on the WDA and CTA from chapter 3.

Chapter 5 discusses the Maple instructional design study and its results. This study tests the application of CLT-based design principles to the set of transferable rules for Maple use.

Chapter 6 discusses the Logo instructional design study and results. The study in chapter 6 tests the use of CLT-based design again, but the participants are tested using problems that involve more distant knowledge transfer.

Finally, Chapter 7 combines the implications of all the studies' results. Contributions and future work are also discussed along with the conclusions.

Chapter 2
Background

This dissertation examines training issues related to the CWA framework. In preparation for this work, this chapter provides an overview of human factors models and focuses on CWA, outlines some current gaps in the literature, and then reviews a subset of the learning literature used later to develop theoretical work around learning as related to the ecological components of CWA.

There are a number of types of models and frameworks used in human factors research. Section 2.1 provides a brief overview of the types of models seen in human factors research and helps frame the role of CWA, the main framework used in this dissertation. Then, in section 2.2, CWA is introduced. The five phases and the relationships between the phases are examined. A major component in the final phases of CWA, and the focus of this dissertation, is the Skill-Rules-Models Taxonomy, which is reviewed in section 2.3. Then the existing research related to training and CWA is examined in section 2.4. This existing training research related to CWA primarily considers the ecological-focused phases of CWA. To start to expand training considerations towards the cognitive end of CWA, the existing work on training literature related to SRM is reviewed in section 2.5, in particular, the meaning of expertise, as described by SRM, is considered. An existing method of developing expertise, the Four Component Instructional Design (4C/ID) method, is also described.

Following the examination of expertise, needed work on identifying training needs is explored in section 2.5.2, along with methods to meet those training needs. The 4C/ID method is one possible method for meeting those training needs, but situations where 4C/ID may be ineffective or too involved are considered.

To begin an examination of how to identify training needs described by SRM, section 2.6 describes some work on training transfer, which is used to guide the theoretical work in Chapter 4. Finally, in section 2.7, CLT is described, which is used in the experiments in Chapters 5 and 6 to develop training materials.

2.1 Types of Human Factors Models

There are a number of types of models that can be potential guides to human factors research. Some models that are useful for the investigation of the interaction of operators and tools and which can be potentially expanded to help consider learning is reviewed here. Given the number of interactions needed to perform tasks in a complex system, along with the previously discussed factors of dynamic events and uncertainty, it is difficult to develop exhaustive models for how operators can or should behave in such systems (Rasmussen, 1985). Hence, models of such systems, and particularly those for operator performance in such systems, should not just focus on descriptions of how known tasks are or should be done, but also should permit the derivation of new patterns of behaviour, either in advance or on the fly, to deal with predicted but yet encountered tasks and unexpected tasks. Vicente (1999) divided human factors models into three categories: descriptive, normative, and formative. Each type of model is important and they need not be used in isolation, nor will any particular model necessarily fit cleanly into a category.

Descriptive models describe how work is actually done in a system (Vicente, 1999). Such models can be based on concepts such as Naturalistic Decision Making (eg. Klein, 1999) and Distributed Cognition (Hutchins 1995). Information for these models can be elicited using methods such as the Critical Decision Method (Klein, Calderwood, and MacGregor, 1989) and Cognitive Ethnography (Hollan et al., 2000). Descriptive models are useful for describing and designing existing systems, but can be of limited use if they describe inefficient ways of performing a task (Vicente, 1999).

Normative models describe how work should be done in a system (Vicente, 1999). Such models are most often derived by examining domain expert behaviour, through methods such as Cognitive Task Analysis (eg. Crandall, Klein, and Hoffman, 2006). Cognitive Task Analysis is used to examine what tasks must be performed in the system, what input and output is expected, how information should flow through the system, and timeline descriptions of the system behaviour (Crandall et al., 2006). Normative models developed with domain-expert input can be problematic as a basis for teaching new users, because a new user might not perform a task the same way an expert does (Vicente, 1999). It is not even guaranteed that all experts would perform a task in the same way (Vicente, 1999). Moreover, a normative model cannot describe the needed behaviour or cognitive support for unknown future scenarios or for a completely new system (Vicente, 1999; Naikar et al., 2003).

Descriptive and normative models will always be important, but the concerns with them leave a need for another type of model. *Formative models* describe the necessary components and constraints of a system, and provide templates for task completion, but allow for flexible descriptions of behaviour in the completion of tasks (Vicente, 1999; Salmon and Jenkins, 2010). Such models do not result in a complete description of the operation of a particular system but define the characteristics of the system that help operators in the system to complete tasks successfully, but the details of operations are left to the operators (Vicente, 1999).

By focusing on constraints to work and not the details of behaviours, formative models allow the development of systems and interfaces that permit operators to be opportunistic and complete tasks in the most efficient way they are able to do so given a system configuration (Vicente, 1999). This flexibility allows operators to use their knowledge to create solutions in new situations, where as systems designed using other types of models may be overly restrictive on the types of solutions an operator could implement (Vicente, 1999). One formative model is CWA (Rasmussen, 1986; Vicente, 1999; Rasmussen, Pejtersen, and Goodstein, 1994), which is the focus model of this chapter.

2.2 Cognitive Work Analysis

Any system that involves humans and technological tools will concern both cognitive factors, related to the humans in the system, and ecological factors, related to the tools and environment (Rasmussen, 1986; Vicente, 1999). Therefore, design of a useful system must take into account the functionality and constraints of both the ecological and cognitive factors of a system (Vicente, 1999). Ignoring cognitive constraints of the operators will result in a system that is not easily used, and ignoring ecological constraints will result in a system that does not have the needed functionality (Vicente, 1999).

Cognitive Work Analysis (CWA) (Rasmussen, 1986; Vicente, 1999, Rasmussen, Pejtersen, and Goodstein, 1994; Naikar, 2012) is a five-phase framework that examines both ecological and cognitive factors in a system. The five phases of a CWA are: Work Domain Analysis (WDA), Control Task Analysis (CTA), Strategies Analysis (SA), Social Cooperation and Organization Analysis (SOCA), and Worker Competency Analysis (WCA) (Vicente, 1999). As a CWA progresses through the phases, the focus of analysis shifts from a focus on ecological factors to a focus on cognitive factors, as pictured in Figure 1 (Vicente, 1999).

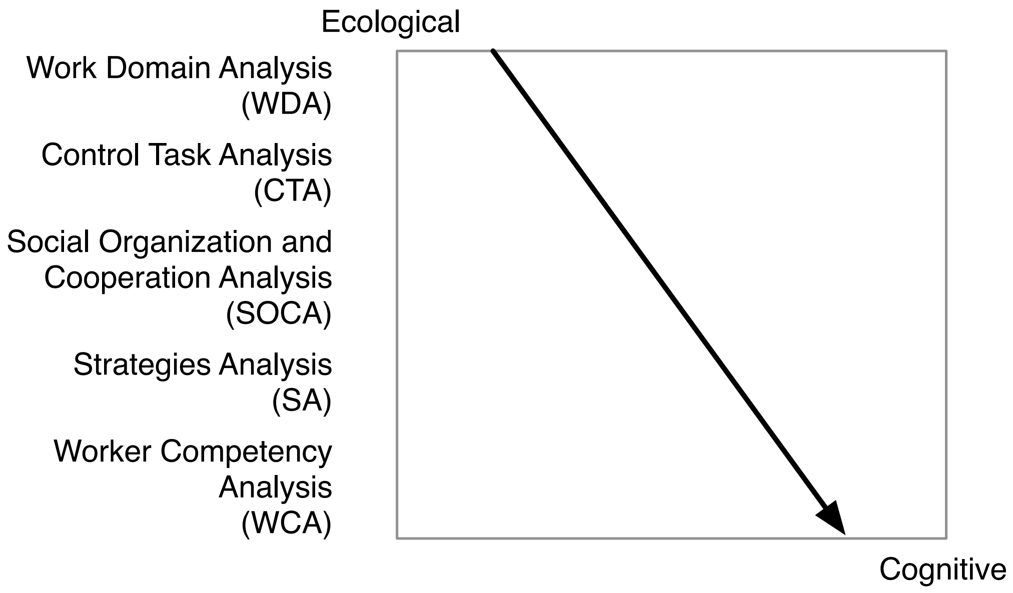

Ecological
Work Domain Analysis (WDA)
Control Task Analysis (CTA)
Social Organization and Cooperation Analysis (SOCA)
Strategies Analysis (SA)
Worker Competency Analysis (WCA)
Cognitive

Figure 1 - Shift from ecological to cognitive factors as phases of as CWA progress

Use of the CWA phases is more of an iterative process than a sequential one (McIlroy and Stanton, 2012). For example, as will be described in more detail, Ashoori and Burns (2012) recommend the re-visitation of previous phases to complete the SOCA rather than treating SOCA as a distinct phase. Furthermore, McIlroy and Stanton (2011, 2012) explained that if there is no benefit of all phases to a particular application of CWA, not all phases need to be completed. They argued that, and gave an example where, the information needed for completing a WCA was collected directly from a WDA. Therefore, while each phase in a CWA provides potentially useful information, the phases can be useful in different combinations and possibly different orders.

Various combinations of the CWA phases have shown up in a number of domains including military domains (Jenkins, Stanton, Salmon, Walker, & Young, 2008; Naikar, Pearce, Drumm, & Sanderson, 2003), air traffic control (McIlroy and Stanton, 2011), road safety (Cornelissen, Salmon, & Young, 2012), medical settings (Ashoori, 2012; Ashoori, Burns, Momtahan, & d' Entremont, 2011; Ashoori, & Burns, 2012; Hajdukiewicz, Vicente, Doyle, Milgram, Burns, 2001; Sharp & Helmicki, 1998), consumer electronics design analysis (Cornelissen, Salmon, Jenkins, & Lenné, 2012), team work analysis (Naikar, Pearce, Drumm, & Sanderson, 2003; Ashoori, 2012), and training program design (Naikar 2006b, 2009; McIlroy and Stanton, 2012).

One particular application of CWA mentioned frequently in this dissertation is Ecological Interface Design (EID) (Rasmussen & Vicente, 1989; Vicente & Rasmussen, 1992; Vicente 2002). EID is a method for designing interfaces that support operator behaviour in complex systems where unknown events are likely to occur, and its design principles are based on SRM and the ecological psychology of James Gibson (Vicente & Rasmussen, 1992; Burns and Hajdukiewicz, 2004). EID is currently one of the most active uses of SRM and CWA, and is a source of many current studies related to the frameworks (e.g. Burns et al. 2008; Drivalou, Marmaras, 2009; Kim, Suh, Jang, Hong & Park, 2012).

2.2.1 Work Domain Analysis

The first phase of the CWA, Work Domain Analysis (WDA), is focused on the objects and concepts in a work domain that the operators in the system have to consider when completing cognitive tasks (Vicente, 1999). A sufficiently complex work domain will be too complex for an operator in the system to understand completely at any one time, due to the number of objects in the system and the number of possible interactions between all the objects (Rasmussen, 1985). Furthermore, constructing technological tools to assist in a complete understanding of the system would result in just as much complexity by the law of requisite variety (Ashby, 1958; Rasmussen, 1985). Therefore, a method for grouping objects is needed. A WDA normally uses Rasmussen's (1985) abstraction hierarchy to analyse those concepts and objects and help the operators form a mental model of the system.

Figure 2 shows a simple example of an abstraction hierarchy for a computer used to solve problems. The number of levels in an abstraction hierarchy can depend on the work domain, the needs of the operators in the system, and the desired resolution (Rasmussen, 1985; Vicente, 1999). Each node in an abstraction hierarchy represents a concept or a physical object in the work domain. The abstraction hierarchy divides up the work domain along two dimensions: decomposition and abstraction (Rasmussen, 1985; Vicente, 1999). The decomposition dimension divides a system into subparts. For example, a computer could be divided into a monitor, a mouse, a tower, and a keyboard. The tower could be further divided into a hard drive, a CPU, an optical drive, random access memory, and a power supply (this example is obviously simplified). The abstraction dimension divides a system by functional purpose and form (Vicente, 1999). Moving up in the abstraction dimension from a node answers why the objects or concepts are in the system and moving down answers how the object or concept is implemented (Rasmussen, 1985; Vicente, 1999).

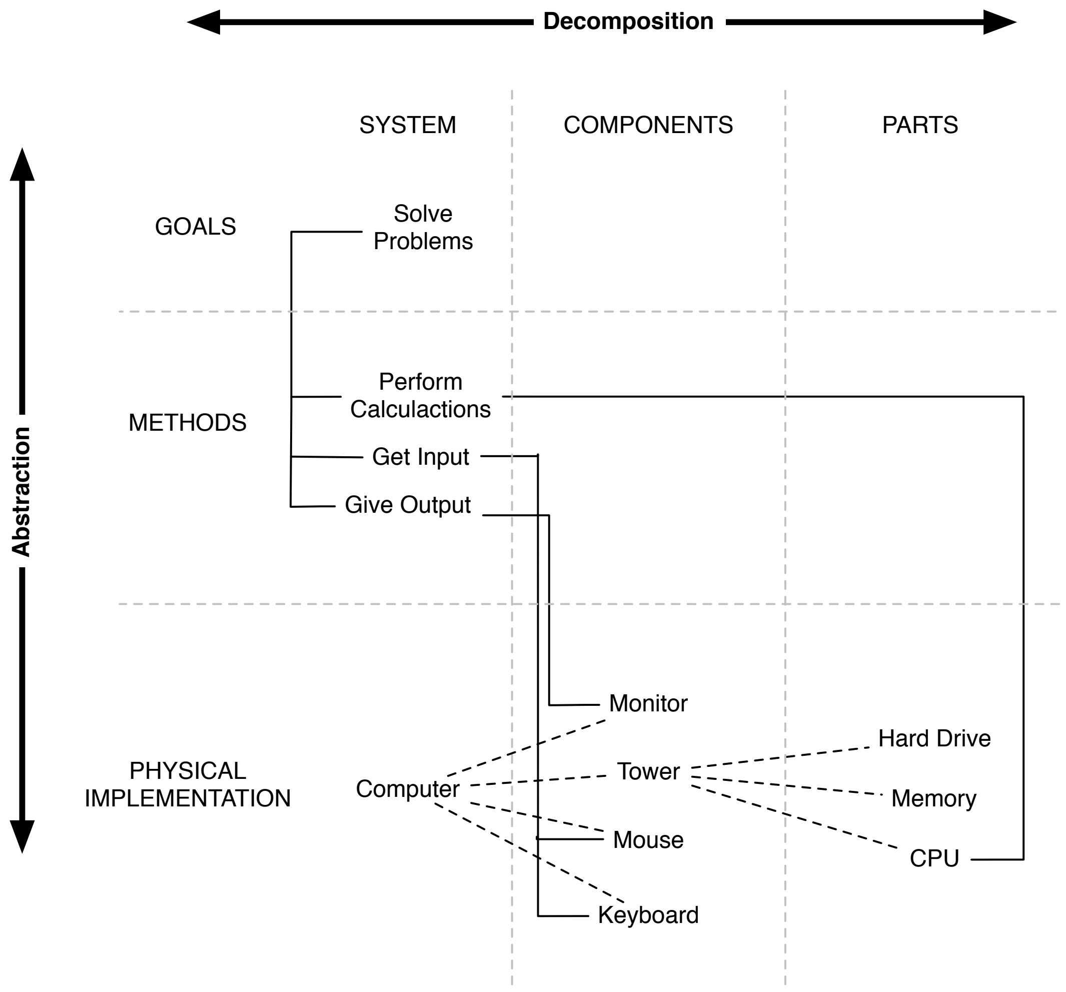


Figure 2 – Simplified abstraction hierarchy for a computer system

An abstraction hierarchy represents a system in a way that can be translated into a more manageable mental model of the work domain for an operator or into a more manageable interface for the operator to use when managing the work domain (Rasmussen, 1986; Vicente, 1999). Furthermore, the abstraction hierarchy can be considered as a tool for functional reasoning: an operator can consider the purposes of a work domain component or concept by moving to a higher level in the abstraction hierarchy or determine why a system is failing to accomplish some purpose by moving down the hierarchy to consider how the purpose is met or right to consider subparts involved (Rasmussen, 1986).

While an abstraction hierarchy can vary in the number of levels in the abstraction dimension (Rasmussen, 1985; Vicente, 1999), most abstraction hierarchies have five levels (Rasmussen, 1985; Rasmussen et al., 1994; Vicente, 1999; Burns & Hajdukiewicz, 2004). The names and descriptions of the levels vary from publication to publication. This dissertation uses the definitions from Vicente (1999) and Burns and Hajdukiewicz (2004).

The upper two levels of the abstraction hierarchy describe the purposes of the system. The top level, the *functional purpose,* describes the reasons of the design of the system (Rasmussen et al., 1994). The second level from the top, the *abstract function,* describes the causal structure of the system and the constraints that must be met to achieve the purposes of the system (Burns & Hajdukiewicz, 2004; Vicente, 1999).

The lower two levels of the abstraction hierarchy describe the form of the components in the system and their functions. Starting at the bottom, the lowest level in the abstraction hierarchy is the *physical form* (Vicente, 1999). The concepts at this level represent the characteristics of the objects in the system, including the tools, equipment, and their descriptions (Burns & Hajdukiewicz, 2004). These physical objects afford the functional capabilities represented at the next level in the abstraction hierarchy, the *physical functions* (Vicente, 1999; Burns & Hajdukiewicz, 2004).

The *general functions*, the middle level of the abstraction hierarchy, links the two lower, physical levels, to the upper, system-purpose levels (Jenkins et al., 2010). The general functions level described the functions and processes of the system that are used to achieve the abstract functions of the system using the objects and processes found in the lower levels (Vicente, 1999; Rasmussen et al., 1994).

The goal of a WDA is to identify the objects and concepts in the work domain, their purposes, and their relations to other concepts in the work domain. Furthermore, a WDA can also be important in describing the operating constraints of various components in the system (Vicente, 1999; Naikar, 2009). In some situations, the information from a WDA is quite useful for designing an interface, such as in the EID process found in Burns and Hajdukiewicz (2004); however, in other cases the interaction between operators and the objects in a work domain needs to be considered in more detail, a task accomplished by the second phase of CWA, Control Task Analysis.

2.2.2 Control Task Analysis

The Control Task Analysis (CTA) phase begins the analysis of interaction between operators in a system and the work domain (Vicente, 1999). A WDA considers the objects in a system in a stateless form, but during the operation of a complex system, the objects in a work domain will often be in a particular state that can be defined by an number of attributes such as if the object is on or off, or its power output (Hajdukiewicz & Vicente, 2004). Therefore, in connecting WDA to CTA, Hajdukiewicz & Vicente (2004) recommended the consideration of work domain state. The operating of a system will require completion of tasks to move the objects in a work domain from an initial state to a goal state (Hajdukiewicz & Vicente, 2004), such as the turning on of a device or increasing its output. CTA is intended to examine what are those tasks that need to be completed in a work domain, but without consideration of how the tasks are completed or by whom (Vicente, 1999; Naikar, 2006a; Hajdukiewicz & Vicente, 2004). A method for CTA needs to help determine possible interactions with a work domain when operators work flexibly in sometimes-unexpected situations, to meet the flexibility goal of CWA.

Rasmussen (1976, 1986) proposed the decision ladder as a template of interaction between operators and a work domain. A generic decision ladder is shown in Figure 3. The round nodes represent states of knowledge (Vicente, 1999) and the rectangular nodes represent information-processing activities that transform one state of knowledge into another (Rasmussen et al. 1994). Lintern (2009) grouped the various states of knowledge and processing tasks into the broad categories of analysis, evaluation, and planning.

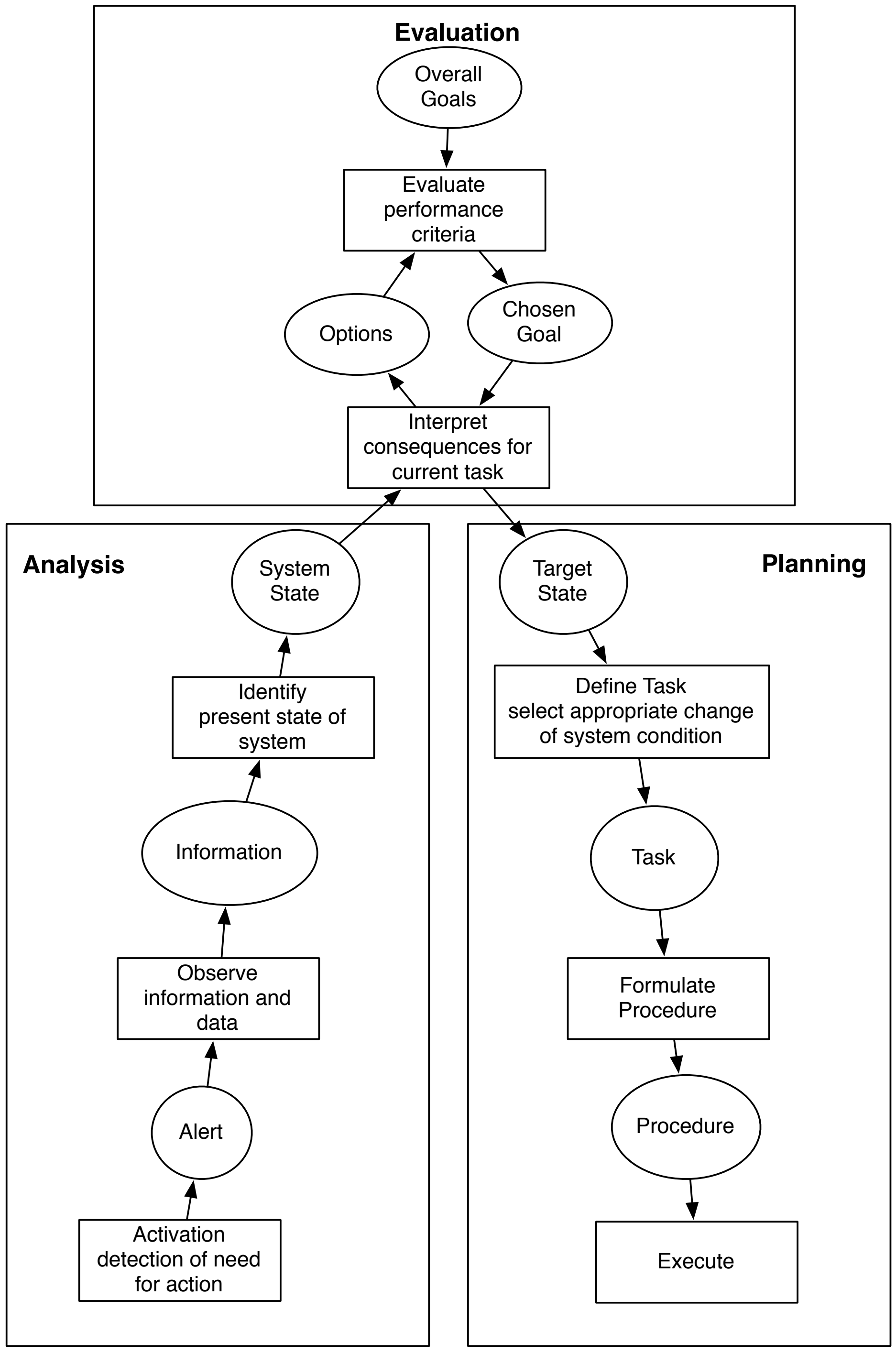

Evaluation
Overall Goals
Evaluate performance criteria
Options
Chosen Goal
Interpret consequences for current task
Analysis
System State
Identify present state of system
Information
Observe information and data
Alert
Activation detection of need for action
Planning
Target State
Define Task select appropriate change of system condition
Task
Formulate Procedure
Procedure
Execute

Figure 3 - Generic decision ladder. Rectangles represent information-processing tasks and circles represent states of knowledge. Adapted from Elix and Naikar (2008).

While the presentation of the decision ladder implies a linear process of moving from activation to execution, representing a very formal decision making process (Rasmussen, Pejtersen, Goodstein, 1994), real operator behaviour often does not follow the formal path, and it can be difficult to know exactly what path an operator takes. As real operators gain experience, they take shortcuts when they recognize situations similar to those they have previously dealt with (Rasmussen et al., 1994; Rasmussen, 1986). On the other hand, in unfamiliar situations, the behaviour at individual information-processing activities, in some cases, could be modelled as activity in a sub-decision ladder (Lamoureux & Chalmers, 2009). Furthermore, even knowing what path an operator takes can be difficult, because, as Rasmussen (1976) noted, during the studies that led to the decision ladder, study participants were often unable to verbalize why they made a decision; they often just knew what to do. This inability to verbalize could be due to Vicente's (1999) claim that during rule-based behaviours, operators do not explicitly consider goals or reasons for actions. He also claims that, in such cases, if the operators can explain their actions later, the explanation is a rational reconstruction of reasons. Moreover, as Rasmussen and Olsen (1989) noted, as people gain expertise, the ability to state reasons for actions can become difficult. Because of these complexities in real operator behaviour, the decision ladder cannot be thought of as representing a description of a sequential decision process.

Rasmussen (1986) called the shortcuts that can occur in a decision ladder leaps and shunts. Shunts are connections from an information-processing task to a knowledge state and leaps are connections between two knowledge states without an intervening processing activity (Rasmussen, 1986). Figure 4 shows a decision ladder with a leap and a shunt shown. A leap might occur when a frequent alert is seen that an operator knows the appropriate response for, and a shunt might occur when familiar observations are associated with a task that needs to be completed. It is also possible for a person to move from the right side of the ladder to the left, say from the task node to the observe node, to collect more information from the environment (Rasmussen, 1986). Rasmussen (1974) noted that operators often cannot verbally express reasons for those shortcuts.

Lintern (2009) questioned the appropriateness of leaps and shunts, as he claimed they are too suggestive of cognitive processes or of defining how a task is accomplished. Lintern suggested sticking to the idea of state transitions, but he did not specify if those transitions allow movement around the ladder in a non-sequential manner. However, Vicente (1999) claimed that consideration of non-sequential transitions is a useful method of using the decision ladder in a formative manner to determine the information and cognitive supports that would be necessary should an eventual operator decide that making such a leap or shunt would be useful and possible. In that case, the use of the decision ladder to consider cognitive support means that some consideration of cognitive process at the CTA phase of CWA is unavoidable because the possibility of leaps and shunts requires certain cognitive abilities.

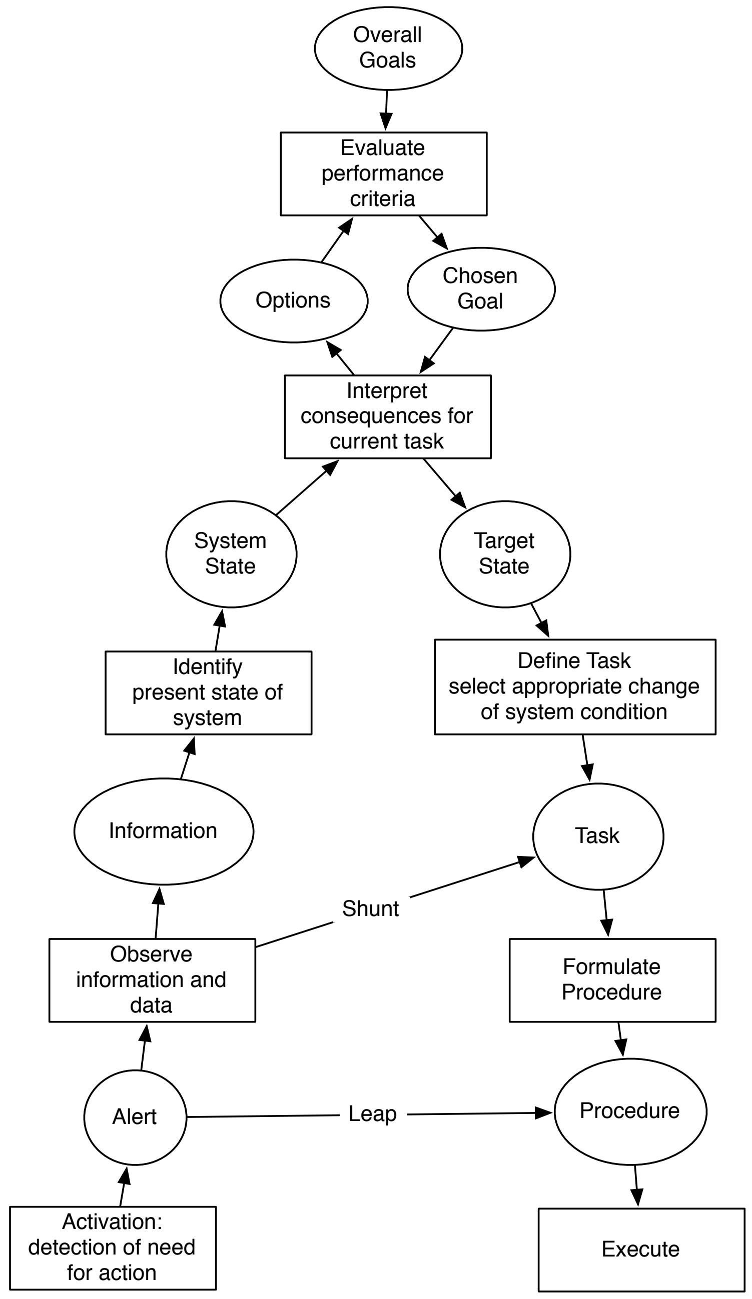

Overall Goals
Evaluate performance criteria
Options
Chosen Goal
Interpret consequences for current task
System State
Target State
Identify present state of system
Define Task select appropriate change of system condition
Information
Task
Shunt
Observe information and data
Formulate Procedure
Alert
Leap
Procedure
Activation: detection of need for action
Execute

Figure 4 - Decision ladder with leaps and shunts

In consideration of the relation between cognition and the decision ladder, Rasmussen (1986) stated that the ladder should not be thought of as a model of cognitive activity but as a map of possible states of knowledge and information-processing activities that can be used to identify possible information requirements of an operator and cognitive shortcuts that operators could potentially employ, if they were given appropriate supports to do so (also Vicente, 1999). However, Rasmussen (1976) did refer to the decision ladder as a model, and Lintern (2010) claims the decision ladder represents a comprehensive model of cognitive states, at least in the "loose sense in which the term model is used in cognitive engineering" (p. 306). This distinction is not essential to resolve here, but it is worth noting that different authors take a different view of the nature and use of the decision ladder.

Elix and Naikar (2008) gave a process for using the decision ladder to analyse or model how decisions might be made in a system. When operating a system there are a number of system goals that must be achieved, and these goals can include the respecting of system constraints (Elix and Naikar, 2008). The system is operated by changing the state by completing tasks to achieve the stated goals. Elix and Naikar (2008) examine the meaning of the components of the decision ladder by examining the questions that a person must be able to answer at each knowledge state. Table 1 summarizes their claimed meanings of the states.

Table 1 – Descriptions of information states in the decision ladder. (From Elix and Naikar, 2008)

State	Purpose in analysis
Alert	The recognition of a need to act or complete a task. Represented by cues that might indicate the need to take action.
Information	Set of observations or data that represents the state of a system. May be derived from consideration of information needed to complete a task and information presented by a system that may be useful.
System State	The current state of the system. Based on a situation assessment relevant to goals.
Options	Possible actions that may help in achieving overall goals.
Overall Goals	The overall goals in operating a system. Guides all other processes in the decision ladder.

State	Purpose in analysis
Chosen Goal	The goal that will be achieved by an action. Many goals might interact with one another, but this state represents a choice of a goal to act on at a particular moment. Considerations can include how choice of one goal might affect other goals.
Target State	The state that will result from the chosen option.
Task	The task that needs to be completed to achieve the chosen target state.
Procedure	The procedure that can be executed to complete the task and achieve the target state.

For an example of how a decision ladder might provide a mapping of how a person performs a cognitive task, consider the computer example from the WDA section. Figure 5 shows the control task that might occur as a result of a computer not turning on. The example assumes the operator is fairly experienced with the computer. First, the operator would notice the computer not turning on as per his expectations, generating an alert. Then the operator might make further observations to see if there is any response from the computer, discovering no response. If the operator had seen this situation before, he might make the immediate leap to knowing that he should check to see the computer has power. Given that checking power cables to confirm power is a routine task, the operator probably does not need to consider how to check the cables, so he can just execute the checking procedure. After checking the cables, the operator may need to observe more data to decide if the problem is fixed or if more work is needed. In the design of this ladder, it was assumed that the operator is experienced with computers, but an inexperienced operator might need to use more steps, including more explicit consideration of system state and possible solutions.

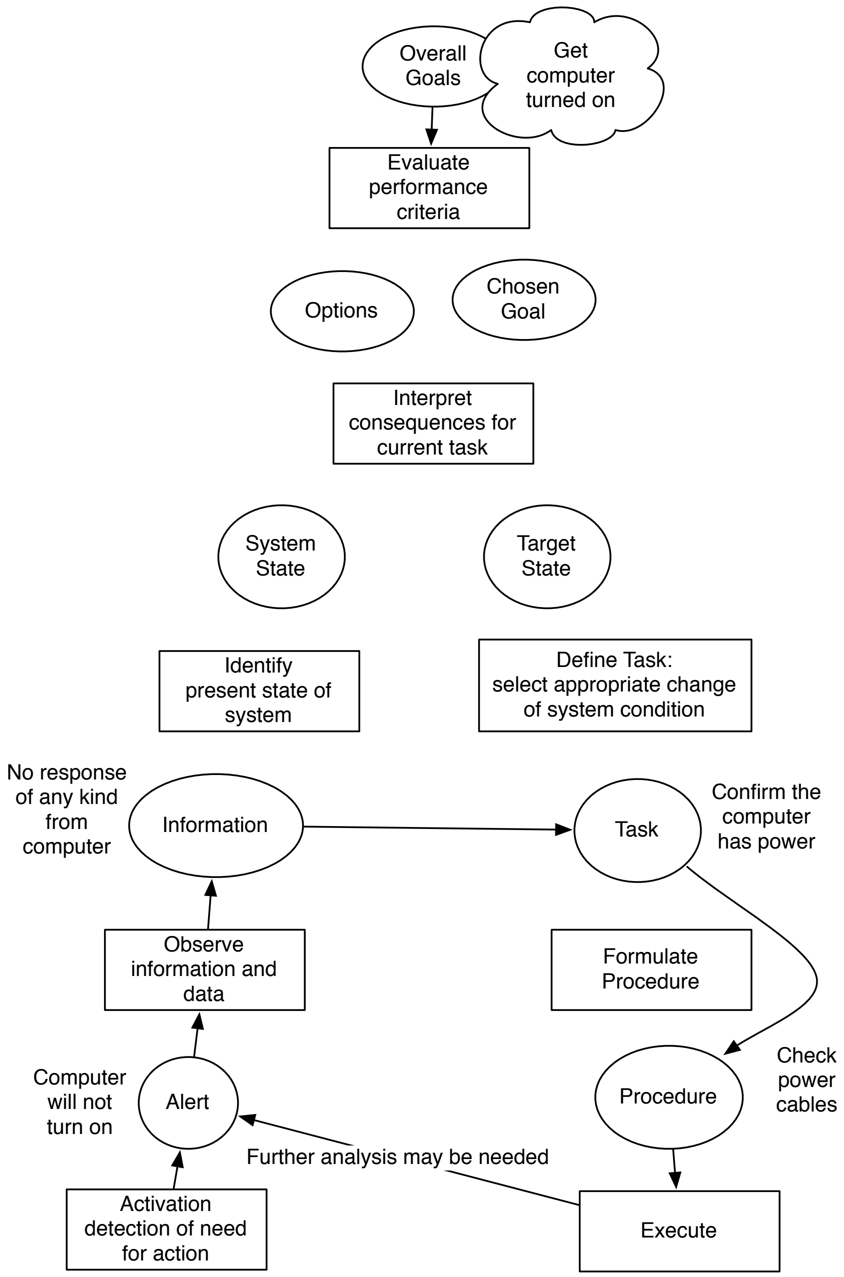

Overall Goals
Get computer turned on
Evaluate performance criteria
Options
Chosen Goal
Interpret consequences for current task
System State
Target State
Identify present state of system
Define Task: select appropriate change of system condition
No response of any kind from computer
Information
Task
Confirm the computer has power
Observe information and data
Formulate Procedure
Check power cables
Computer will not turn on
Alert
Procedure
Further analysis may be needed
Activation detection of need for action
Execute

Figure 5 – CTA decision ladder for diagnosing a computer startup problem

It is possible to consider multiple decision ladders in the analysis of a cognitive activity. Rasmussen, Pejtersen, and Goodstein (1994), use multiple decision latters to analyse multiple tasks in a situation. Naikar, Moylan and Pearce (2006) take this idea further, recommending preceding the use of a decision ladder with a Contextual Activity Template (CAT). The CAT is used to determine in what situations and contexts a task usually does and can possibly occur. Using that contextual information, a decision ladder can then be generated for each situation in which a task occurs.

The decision ladder is a template of tasks that can occur in a control task, but the information-processing steps remain as black boxes. The next phase in a CWA, Strategies Analysis, looks at categories of strategies that can be used for the information-processing steps.

2.2.3 Strategies Analysis

In a Strategies Analysis (SA), more details about the information processing steps in the CTA are filled in (Vicente, 1999), albeit, as will be shown, the details developed do not constitute a set of algorithms for the tasks, for two main reasons. As already noted in the discussion on CTA, different operators might approach the same task in different ways, taking opportunistic leaps of knowledge (Vicente, 1999). Furthermore, it has been noted that in a complex system, operators need to be able to deal with unexpected situations. Therefore, defining a set of algorithms for completing tasks is not possible, because the set of algorithms will be necessarily incomplete and any system or interface designed based on such algorithms will be insufficient in some situations (Vicente, 1999).

Instead of defining algorithms for strategies, Vicente (1999) suggested determining categories of strategies. Vicente likens this setup to having bins of strategies that an actor can choose from, based on the situation and his preferences. It is possible for an actor to change to a different strategy type during a task (Vicente, 1999; Naikar, 2006a). As an example of how a person can use different strategies when completing a task, the computer example from the WDA can be considered. If a computer is not working, a technician can try to determine what component is not working. There are a number of categories of strategies that could be employed, but for brevity two are considered. First, the technician could start the computer and look for an error code that indicates which component is not working. Alternatively, the technician could try various components from the non-working computer on a test computer, until he or she determines which component does not work. Furthermore, it is easy to see when a technician might change strategies. If the technician tries to start the computer, but it will not start such that he can retrieve an error code, the second method, of trying components, may need to be employed.

Strategies such as those just described could come from a number of sources. The most obvious sources are descriptive and normative studies, using methods such as cognitive task analysis or naturalistic observations (Vicente, 1999). Those methods are valid ways of determining strategies; however, using such methods may miss other ways of doing work by other experts or by non-experts. Furthermore, those methods do not completely satisfy CWA's philosophy of being a formative method, and, therefore, will not work for new systems for which experts do not exist. To make up for these shortcomings, Vicente (1999) suggested using a priori methods of examining the work domain and control tasks to develop potential strategies. To this effect, Cornelissen, Salmon, Jenkins, and Lenné (2012) proposed the SA Diagram as a method for determining new strategies. The SA Diagram uses information from the WDA and CTA, in a diagram similar to an abstraction hierarchy, to derive strategies. In contrast, Kilgore, St-Cyr, and Jamison (2009) use a more traditional information flow map to analyse strategies.

Different types of application domains and analysis lead to different types of strategy descriptions. There are a variety of examples of strategies, with differing levels of details, from different authors. For example, Cornelissen et al. (2012), when analysing strategies for troubleshooting problems with an iPod music player derive strategies such as "If the iPod is not charged then connect the personal computer" (p. 16), by using the SA Diagram. Kilgore et al. (2009), after using information flow maps to troubleshoot problems with aircraft separation in an Air Traffic Control task, describe strategies such as "A series of commands are given to adjust one aircraft's flight path in a manner that eliminate a potential loss of separation with the other aircraft, while minimizing the magnitude of change to its original flight path." (p. 32)

In principle, the strategies determined in a SA are operator and tool independent (Vicente, 1999). Once the strategies are determined, any operator, tool, or combination, can be used to implement them. However, in reality some strategies will have to be tailored to the operators or tools implementing them, such as the iPod example just above. For strategies that are operator independent, the task of determining who will implement a strategy occurs during the next phase of a CWA: SOCA.

2.2.4 Social Organization and Cooperation Analysis

The previous phases of CWA were agnostic (in theory) to which operators and tools in system complete a task. The phase for Social Organization and Cooperation Analysis (SOCA) examines how work is divided up amongst the various operators and tools in a system (Vicente, 1999).

While SOCA is in theory the first stage in which allocation to particular operators and tools is considered, in reality there will often need to be some consideration in earlier phases. In CTA, the decision ladder permits non-sequential transitions between cognitive states. As was discussed, such transitions will sometimes be possible only for operators and tools with certain abilities. Likewise, in the SA, some strategies may be achievable only by some operators and tools. Moreover, some authors explicitly suggest returning to the results of the previous phases to add SOCA-related information (eg. Ashoori and Burns, 2012). Ashoori (2012) and Ashoori and Burns (2012) noted the need to explicitly consider the division of work and the social interactions at all phases of a CWA. They suggested that an analyst, during a SOCA, revisit the previous phases of a CWA to add information about work divisions and interactions between various operators and tools to the results from each phase.

There are several examples in the literature of the addition of social information to previous phases. Vicente (1999) looked at the division of a work domain by using a responsibility map. Then Jenkins, Stanton, Salmon, Walker, and Young (2008) examined the allocation of control tasks in various contexts across different operators and tools using a colour-coded contextual activity template. Expanding the work on the social aspects of CTA, Ashoori and Burns (2010) and Ashoori, Burns, Momtahan, and d' Entremont, B. (2011) examined the interactions of different operators' control tasks using the decision wheel.

Regardless of if SOCA is treated as a separate phase or a re-visitation of previous phases, the essence of the phase is to determine the best operator and tool, or combination thereof, in a system to perform each cognitive task (Vicente, 1999; Naikar, 2006a). The role of SOCA should not be confused with the development of formal structures of command, such as those described by organization charts, but SOCA is an examination of the functional organization used to complete cognitive tasks (Rasmussen et al., 1994). These organizations are structured around the functionality of the work domain and may evolve as the work changes or the various operators gain abilities (Rasmussen et al., 1994).

A functional work analysis and division will address issues such as (Rasmussen et al., 1994):

- What is the work to be divided up?
- What are the boundaries of an operator's work?
- What are the tasks that an operator can and needs to complete?
- How do the operators complete those tasks?

Such an analysis will help determine how the work to complete tasks can be divided up amongst operators and tools. In this dissertation, the main issue being discussed is how operators learn to use various tools in a system to complete tasks. Therefore, questions that are most relevant pertain to what tasks the tools in a system can perform, relieving cognitive work from operators, and how operators operate the technology to perform the correct tasks. The use of a piece of technology requires two-way communication between operators and tools, fulfilling the observation that cognition is a process of reciprocity with the objects in the environment. Therefore, in order for operators to make use of technology to complete cognitive tasks, the technology must provide functionality, cues concerning that functionality, and cues about progress in completing tasks.

2.2.4.1 Affordances

For an operator to make use of a tool in a system to complete a cognitive task, the tool must provide some functionality useful to an operator. The match between a technology's abilities and an operator's ability to make use of the technology is what J. Gibson (1977, 1979) referred to as an affordance. J. Gibson (1977) defined an affordance as "a specific combination of the properties of its substance and its surfaces taken with reference to an animal". The definition of affordance has drifted, as will be shown, but a useful definition that is a bit broader but still true to J. Gibson's (1977) original meaning is by E. Gibson and Pick (2000). E. Gibson and Pick state, "an affordance refers to the fit between an animal's capabilities and the environment supports and opportunities (both good and bad) that make possible a given activity." (p. 15).

As a side note, there have been a number of definitions of affordance, some more in line with Gibson's (1977) meaning and some nearly the opposite. According to Torenvliet (2003), Norman (1988) added the idea of successful perception to the definition of the concept of affordance, referring to actual and perceived capabilities, diluting the meaning. Torenvliet further claimed that Cooper, Reimann, and Cronin (2007), further adjusted the definition, resulting in the opposite meaning of Gibson, with their definition of affordance referring only to perceived capabilities. J. Gibson's (1977) original meaning of affordance did not include the need for an operator to actually perceive the affordance, as some of these more recent interpretations imply. For example, even if a person cannot find a door handle in the dark, the handle still exists and the door still affords opening, to someone who can find the handle.

If an affordance is an opportunity for the use of a tool to accomplish some task, successful use of the tool requires that it communicate with an operator the opportunity to use the tool and results from using it (Lintern, 2000). In other words, the tool has to provide the operator with appropriate cues.

2.2.4.2 Symbols, Signs, and Signals

A tool in a system can communicate with an operator in a number of ways, but in each case, the tool is providing a cue to its state and, perhaps, the state of related objects in the environment, including other operators. For example, the tool might provide a numerical or textual result from a computation; it might give an alert, such as a dialog box or an audio tone; or it might show a graph indicative of an on-going process' state. There are a limitless number of cues that a theoretical tool can provide and an operator might find different types of cues useful in different situations, but Rasmussen (1983) gave a categorization where he divided the cues provided by a system into three categories: symbols, signs, and signals.

Symbols are cues that have some inherent, meaningful information content about the state of the system that can be interpreted and comprehended by an operator (Rasmussen, 1983; Vicente, 1999). In a sense, operators can use a content of a symbol to figure out what the meaning of the symbol is, but they may still need to use some pre-existing knowledge to do so. For example, if a computer program displays an error dialog, an operator might read the error message and deduce from the message, and from knowledge of the program, what the cause of the problem is.

Signs are cues that indicate something about the state of the system, but have no inherent information content (Rasmussen, 1983; Vicente, 1999). The meaning of a sign is determined by an operator's existing knowledge. For example, an alarm might remind an operator to adjust a setting, but the alarm, which is a sign, has no inherent information content with which to reason about the system state.

A *signal* is a piece of, often time-space continuous, information perceived and interpreted without conscious thought (Rasmussen, 1983; Vicente, 1999). Again, a signal has no inherent meaningful content. For example, a person familiar with a computer mouse system who is moving a mouse pointer to a particular point on the screen tracks the location of the mouse cursor relative to the desired destination as a signal and moves the mouse without consciously considering the connection between the mouse cursor location and needed mouse movement.

First, the category into which a cue falls depends on the way the cue is perceived by an operator, not on the presentation or any inherent meaning of the cue (Rasmussen, 1983). What category a cue falls into will depend on the knowledge, expectations, and intentions of an operator (Vicente, 1999). Furthermore, as previously noted in the discussion on the decision ladder, a person could return to observation if more data is needed (Rasmussen, 1986), in which case a cue interpreted in one category could be reinterpreted in another category, if knowledge or needs change. For example, a dialog might be interpreted as a symbol and analysed for its meaning by one operator, but to another operator, the same dialog might be a sign to take some action without any need for analysis. Furthermore, an error dialog might be interpreted as a sign, but on noticing something unusual about the state of a system, the operator might reinterpret the dialog as a symbol and begin to process its meaning.

In a complex system, the distributed and social nature of cognition imposes a need for communication between tools and operators. Cues allow operators to know what a tool affords them and the state of the tool and its environment. How operators interpret the cues and how they know how to act on the tools' affordances, given certain cues, depends on the operators' knowledge for interpretation, possibly along with other information in the environment. Therefore, it is necessary to determine what knowledge and abilities the operators in a system need to complete tasks efficiently using the tools available to them and the cues supplied by those tools. In the next section, covering the final phase of CWA, the abilities needed by workers for using tools in a system are examined.

2.2.5 Worker Competency Analysis

The final phase in CWA, and the one closest to the cognitive end of the ecological-cognitive spectrum, is the Workers Competency Analysis (WCA) (Vicente, 1999). Because WCA is more cognitive in nature, albeit still with consideration of the ecological nature of tasks, it is closer to traditional human factors cognitive task methods (Vicente, 1999). The essence of WCA is to determine what abilities the operators in a system need to perform the strategies determined in the SA phase of a CWA, how operator abilities can be better supported by the tools in a system, and how cognitive constraints of the operators affect the system (Kilgore, St-Cyr, and Jamieson, 2009). Even outside a CWA, as McIlroy and Stanton (2012) pointed out, the WCA can be useful for determining the abilities needed to implement strategies derived from other sources.

When used in the context of a CWA, the analysis of operator abilities will be shaped by the results of the previous phases, particularly SA and SOCA (Vicente, 1999). As already mentioned, operators need the knowledge necessary to process cues from the tools and act on the appropriate affordances. Furthermore, the knowledge that operators have will affect how they perceive the cues, whether they are perceived as symbols, signs, or signals, and how they interpret the meaning of those cues (Rasmussen, 1986).

The results of a WCA are a description of operator abilities that must be possessed or developed, and possibly supported through improved cues, so that the operators can complete tasks effectively in the kinds of complex systems targeted by CWA (Kilgore, St-Cyr, and Jamieson, 2009). Therefore, the results of a WCA need to describe operator abilities that match the CWA philosophy of being formative and flexible to meet complex, and possibly unexpected, situations. So, a WCA needs to have a method for determining yet-unseen abilities needed for operators to develop and implement novel strategies (Kilgore et al., 2009; Cornelissen et al., 2012).

One obvious way to determine the abilities that operators have to interpret cues and act on those cues would be to perform a descriptive or normative study of expert or novice operators. Such a method is certainly valid and would provide useful information. However, as far as the philosophy of CWA is concerned, such methods would be incomplete, because they would not fulfil CWA's role as a formative method of analysis. Such methods would not reveal how operators could function in unanticipated scenarios, and, as Vicente (1999) noted, operators may use tools in novel ways in new or unexpected scenarios. Furthermore, Naikar, Pearce, Drumm, & Sanderson (2003) suggested using CWA for developing new systems. In such applications, there would be many scenarios where there would potentially be no experts who could describe their work. Therefore, Naikar et al. (2003) then suggested using analytical methods for deriving potential worker competencies. Rasmussen's (1983) SRM taxonomy is designed to provide flexible descriptions of operator behaviour and is currently the main tool for WCA.

2.3 SRM Taxonomy

The Skills-Rules-Models (SRM) taxonomy (Rasmussen, 1986, 1983) was developed as a way to categorize types of human behaviour when a human is using a piece of technology. SRM divides behaviour into the categories of skill-, rule-, and knowledge- based behaviour. The taxonomy can be used as a method of organizing the descriptions of behaviours needed to implement strategies developed during an SA (Vicente, 1999), though the use of SRM taxonomy is not limited to CWA.

As will be shown, the SRM categories can be used for the derivation of new behaviours used in implementing strategies.

In the remainder of this paper the terms skill- and rule- based behaviour are used interchangeably with the terms skills and rules respectively, where the various terms are useful for readability and clarity, as is the tradition in the literature (see Rasmussen, 1983, 1986; Vicente, 1999 for examples), but all the terms refer to the behaviour types described by Rasmussen's SRM taxonomy.

Sanderson and Harwood (1988) noted that the roots of the SRM taxonomy appeared in a 1969 paper written by Rasmussen; however, the terms skills, rules, and knowledge (alternative term for model) first appeared in a 1979 report. The original purpose of the framework was to understand and reduce human error (Sanderson & Harwood, 1988). In this sense, the taxonomy was originally intended to provide distinctions into which normative descriptions of behaviour could be fit (Sanderson & Harwood, 1988). Appropriately, Reason (1990) used the taxonomy extensively in his framework for analysis of human error. However, the distinctions of behaviour types have also proved useful in the categorization of interface displays that support the different behaviours and in consideration of training (Sanderson & Harwood, 1988). As a result of SRM's generality, the taxonomy has been used in a variety of application areas, including nuclear plant controller design (Lin, Yenn, & Yang, 2010), medical training (St. Pierre, Hofinger, Simon, & Buerschaper, 2011; Dankelman, 2008; Dankelman, Wentick, Grimbergen, Strassen, & Reekers, 2004; Wentick, Stassen, Alwayn, Hosman, & Stassen, 2003), aircraft maintenance (Hobbs and Williamson, 2002), analysis of visual interaction (Pohl, Sumc, and Mayr, 2012), and other general training scenarios (Hockey, Sauer, & Wastell, 2007). The SRM taxonomy is also used extensively in EID (Rasmussen & Vicente, 1989; Vicente & Rasmussen, 1992; Vicente 2002).

The SRM taxonomy is usually called the Skills-Rules-Knowledge framework, but I find the use of the term knowledge problematic. The term knowledge can have a broader meaning that encompasses any information a person might hold in their long-term memory (e.g. Haskall, 2001), even the knowledge needed for rule-based behaviour. As an alternative, Rasmussen (1986), himself, stated "the [knowledge-based] level might more appropriately be called model-based" (p.102). In the definition of the framework, Rasmussen has suggested that the operator using this level of behaviour is applying some reasoning method to a mental model of a system to determine an appropriate course of action. Therefore, I have adopted the name "Skills, Rules, Model" (or SRM) to clearly indicate the types of knowledge being used.

It should be made clear from the outset that the SRM taxonomy is not designed to be a comprehensive and predictive psychological theory but a categorization of behaviour (Sanderson & Harwood, 1988). The SRM taxonomy describes how people behave but does not explicitly describe why they behave in a particular manner or how they accomplish such behaviour, though throughout the literature, there are references to the cognitive form of the rules and mental models that operators use (Rasmussen, 1983, 1986; Sanderson & Harwood, 1988; Vicente, 1999). Rasmussen (1983) claimed the taxonomy describes the types of behaviour in a qualitative manner that is flexible in order to describe behaviours that can be used in new, unexpected scenarios.

As mentioned, the taxonomy divides human behaviour into three categories or levels of behaviour: skill-based, rule-based, and model-based (Rasmussen, 1983). The various type of behaviour appears in a person's work when making an observation, making a decision, choosing an action, or implementing the action, depending on their expertise and preferences in regards to that decision or action (Rasmussen, 1983, 1986). Each of the behaviours is related to one of the signals, signs, and symbols category of cues previously discussed (Rasmussen, 1983), and each type of behaviour is triggered by the perception of one of the types of cues (Rasmussen, 1983). The skill- and rule- based behaviours are related in that they both occur in a person's actions when he has pre-existing knowledge about how to behave given certain information. The difference between the types of behaviour at the two levels is the automaticity of the response.

2.3.1 Skill-Based Behaviour

Skill-based behaviour, the first level of the SRM taxonomy, describes an action that is a result of a tight sensory-motor coupling between the operator and the environment and is performed without any conscious control, except perhaps an initial consideration of an intention or goal (Rasmussen, 1983). This type of behaviour is a reaction to the perception of a signal type of cue, where the operator perceives a continuous signal that can be responded to without conscious thought (Rasmussen, 1983). Therefore, skilled-behaviour might be considered the result of a number of well-practiced, sensory-motor routines chained together in a smooth action (Olsen and Rasmussen, 1989).

As Rasmussen (1983, 1986) and Vicente (1999) have noted, skill-based behaviour is a well-practiced behaviour and therefore performed automatically (Shiffrin & Schneider, 1977; Schneider & Shiffrin, 1977), without conscious thought, leaving cognitive resources free so the person can concentrate on other tasks.

2.3.2 Rule-Based Behaviour

Rule-based behaviour, the second level of the SRM taxonomy, occurs in familiar situations where a person consciously executes a known routine stored during previous experiences (Rasmussen 1983, 1986). Rasmussen (1983) noted that procedures may be chained together to reach a goal [though the goals are not often explicitly considered (Vicente, 1999)], and Rasmussen et al. (1994) stated that stored procedures can be tailored to particular scenarios encountered. Rule-based behaviour is related to the perception or processing of cues as signs (Rasmussen, 1983; Vicente, 1999). A particular sign triggers a rule, or chain of rules, that determine how an operator reacts to the sign. In reverse, a cue can only be interpreted as a sign if an operator has the knowledge and expectations needed to recognize the cue as the sign (Rasmussen, 1983).

According to Vicente (1999), rule-based behaviour is similar to skill-based behaviour in that it does not require extensive reasoning; however, some conscious processing occurs in the form of an "if-then" rule or stored procedure that the person already has in long-term memory (Rasmussen, 1983; Vicente, 1999). While rule-based behaviour is less automatic than skill-based behaviour, they are both less cognitively demanding than model-based behaviour (Vicente, 1999).

2.3.3 Model-Based Behaviour

Model-based behaviour, the third level of the SRM taxonomy, occurs in unfamiliar situations when no stored rules, skills, or chains of rules and skills fit the given situation (Rasmussen 1993, 1986) and is performed by reasoning about an explicit model of the work domain (Vicente, 1999). This type of behaviour is associated with symbols. Because symbols have some inherent meaning, that meaning, along with pre-existing knowledge, can be used by the operators to reason about the state of the work domain and what actions they need to take (Rasmussen, 1983; Vicente, 1999).

During model-based behaviour, goals are explicitly considered and various paths to obtaining the goals are considered and selected from, guided by an understanding of the system or through trial and error (Rasmussen, 1983, 1986). Model-based behaviour is essentially a problem solving process because the goals are known but a solution for achieving those goals is not (Rasmussen, 1986). Because it is a problem solving process, requiring the maintenance of goals and current state, model-based behaviour is very cognitively demanding (Vicente, 1999).

2.3.4 Interaction of Behaviour Levels

As a point of clarification, a task is rarely completed entirely within one SRM level. Vicente (1999) noted that interactions between the levels occur. Rule- and skill- based behaviours can be used as part of model-based behaviour activity when recognized states are encountered during a problem-solving process (Vicente 1999). Vicente further noted that actions on the world occur at the skill-based behaviour level, because, ultimately, any physical activity, such as hand movement or sound utterances, must occur at that level.

The combination of behaviours has a benefit to operating a system correctly and efficiently. Skill- and rule- based behaviours impose a lower cognitive demand, leaving cognitive resources free for other tasks (Rasmussen, 1986; Vicente, 1999). Additionally, such behaviours can be performed more quickly (Vicente, 1999), and with fewer errors (Reason, 1990).

Rasmussen (1983) and Reason (1990) have both pointed out that operators will tend to use rule-based behaviour whenever they can, and they sometimes use a known rule even in situations where it is inappropriate. To avoid such errors, operators must know when they do not have the ability to perform a task using skill- or rule- based behaviour, due to an unfamiliar situation, and that they must switch to model-based behaviour in order to complete a task correctly (Rasmussen, 1986).

The above shows that completion of most tasks will require a variety of behaviours of different types, so a variety of possible behaviours needs to be identified.

2.3.5 What Skills, Rules, and Models does an operator need?

In order to use the SRM taxonomy in the design of a system, it is necessary to identify the various skill-, rule, and model- based behaviours that an operator can use. These behaviours will allow an operator to interpret the cues from tools in the system (Vicente, 1999), know how to act to accomplish tasks (Rasmussen, 1983), and, therefore, implement the strategies identified in the SA phase of a CWA. The behaviours identified can also be related back to the decision ladder in the CTA phase, as model-based behaviour will be involved in the evaluation of options and goals, and rules and skills will permit the leaps and shunts that can occur in the ladder (Rasmussen, 1986; Rasmussen et al., 1994; Lintern and Naikar, 1998).

Once important behaviours are identified, the tools in a system can be modified to provide better affordances and cues to support the behaviours used (Olsen and Rasmussen, 1989). Furthermore, new operators can then be trained to use the particular behaviours supported by the tools (Olsen and Rasmussen, 1989), which is the issue explored in this dissertation.

The descriptions of the behaviours, once developed, must still allow for the complex, and sometimes unpredictable, behaviour of operators in complex systems. As has been discussed, operators may mix the types of behaviours, chain together rules, and even switch strategies to reduce cognitive effort and complete tasks more efficiently or to try to find methods to complete novel tasks. Operators may also implement behaviours that are related to different levels of the abstraction hierarchy as identified in a WDA (Rasmussen, 1986). Model-based behaviour is likely to span levels of the abstraction hierarchy, because an operator, when using model-based behaviour, is likely to consider the purpose of and implementation of various concepts and objects in the system (Rasmussen, 1986). However, rule-based behaviour could also be related to various levels in the abstraction hierarchy (Rasmussen, 1986; Olsen & Rasmussen, 1989). For example, considering the computer abstraction hierarchy of Figure 2, an operator trying to compute the answer to a math problem could consider the rules for determining the input needed at the methods level and then the actual keys that he or she needs to press at the physical implementation level. Given all these considerations, methods are needed for determining the flexible sets of behaviours operators might use in a system. The consideration of behaviours at multiple levels of an abstraction hierarchy is also part of EID design, because, as part of the EID design process, relations between objects and concepts, as well as constraints, in the abstraction hierarchy are used to generate visual representations that support the various types of behaviours (Burns & Hajdukiewicz, 2004).

The most obvious method for determining behaviours for accomplishing tasks is to examine what experts do through descriptive and normative studies (Olsen & Rasmussen, 1989). However, using such methods introduces the same issues as they do in SA: the methods may miss some potentially more efficient behaviour and they will not work for new systems. Even when trying to determine behaviours already used in a system, it can be difficult to extract information from experts about why they use certain rules and, even more so, about skills, because as operators learn rules and skills, their knowledge of the explicit goals and cues that trigger the behaviours might not be explicitly learned (Rasmussen, 1986; Vicente, 1999). Furthermore, even if that information was known explicitly at some point, that knowledge tends to become more tacit as expertise increases (Dreyfus & Dreyfus, 1988; Olsen & Rasmussen, 1989). Accordingly, a more analytical method for determining the behaviours that can be used to implement the strategies is needed.

Kilgore and St-Cyr (2006) proposed the SRM inventory as an analytical method for determining the behaviours that can be employed in using various strategies to complete information-processing tasks. The method behind the SRM inventory is fairly simple. First, a table is constructed with columns for each of skill-, rule-, and model- based behaviours. Then, for each information-processing task, methods of completing the tasks using skill-, rule, and knowledge-based behaviours are filled in through a brain storming process (Kilgore, St-Cyr, & Jamieson, 2009). The information included in the table can included generalized descriptions of the behaviours (i.e. rules may not necessarily be stated as an if-then action) and can describe behaviours at various levels in the abstraction hierarchy, as seen in in Kilgore et al. (2009).

As an alternative method for determining needed behaviours, McIlroy and Stanton (2011) create a SRM inventory directly from a WDA abstraction hierarchy. In their paper, the authors completed the SRM inventory by considering what competencies were needs to achieve the various abstract functions described in the abstraction hierarchy.

Once a set of possible behaviours has been developed, it becomes possible to design better affordances and cues for the tools that support operator behaviour. Vicente (1999) and Drivalou and Marmaras (2009) provided some general recommendations for supporting each of the behaviour types, as summarized in Table 2. Not all recommendations will make sense in all work domains, and, in some cases, some of the supports may help with other behaviour types, which is not surprising, considering the mixing of behaviours that can occur.

Table 2 - Supports for skill-, rule-, and model- based behaviours

	Vicente (1999)	Drivalou & Marmaras (2009)
Skill-Based Behaviour	Provide for direct manipulation of a display.	Layout controls such that they correspond to the physical system Help with orientation of cues across displays Provide for direct manipulation of a display
Rule-Based Behaviour	Provide consistent mapping between the work domain and the signs from the interface.	Show important constraints and affordances in a system Provide direct support for expected rules Help with recalling previous experiences and common situations
Model-Based Behaviour	Provide an abstraction hierarchy	Make system goals and relations visible including - functional relations - structural relations - physical relations

An example of a design paradigm with a strong connection to SRM is EID (Rasmussen & Vicente, 1989). The theory behind EID describes interfaces that provide direct support for the various behaviours of SRM, through methods such as those just discussed in Table 2 (Vicente & Rasmussen, 1992). Burns and Hajdukiewicz (2004), for example, provide a number of EID-based controls that provide cues that can be processed as needed by operators using the different SRM behaviour types. By providing support for different behaviours, EID displays accomplish the two goals of encouraging use of the more cognitively efficient rule- and skill- base behaviours when possible but still supporting model-based behaviour when needed (Vicente, 2002).

Once the different behaviours have been described, and possible support for those behaviours considered, it will be necessary in some cases to teach operators to use the behaviours. New operators will have to be taught the behaviours. Additionally, if newer, more efficient behaviours were discovered in the processes of a CWA, it would also be desirable to teach expert operators new behaviours. In the remainder of this section on CWA, the existing work on learning behaviours is examined, but as will be shown, this work is limited, and more work remains to be done.

2.4 Use of CWA in Training

Naikar (2006b, 2009) pointed out that the consideration of training is a critical component of complex system design, particularly when designing new systems where the cost of training must be factored into the cost of such systems. In the consideration of training for new systems or unanticipated scenarios, training design cannot rely on the existence of experts from whom training procedures can be determined via descriptive or normative studies (Naikar, 2009). Therefore, CWA, and its formative philosophy, can play a role in the design of training in such scenarios (Dainoff, Mark, Hall, Richardson, 2000).

Naikar (2006b, 2009), Lintern and Naikar (2000), McIlroy and Stanton (2012), and Dainoff et al. (2000) considered the use of a WDA and abstraction hierarchy in the design of training protocols. In a new system or for unknown scenarios, training of exact procedures is impossible; however, if an operator has an understanding of a system at various levels of the abstraction hierarchy, in particular an understanding of the limits of reasonable or safe operation of a system, he or she is more likely to be able to function successfully in a system and recover from errors (Vicente, 1999; Naikar, 2009). A WDA describes the nature of a system and how and why the system functions as it does, so any training protocol must be based on the information that could be described in a WDA (Naikar and Sanderson, 1999). Naikar and Sanderson (1999) explored the relevance of each level of a WDA to training and how the means-ends and part-whole links highlight training needs in terms of operators learning opportunities for, and limitations of, actions in the system. To this end, Naikar (2006b, 2009) suggested several uses of the abstraction hierarchy in the design of training protocols, including scenario generation, derivation of functions and purposes of the system, understanding of the physical implementation of the system, measures of performance, and data collection methods.

Naikar (2006b, 2009) and Naikar and Saunders (2003) also looked at the use of CTA and the decision ladder in the design of training protocols. They looked at critical incidents where operator error could occur and considered each information-processing step in the decision ladder in such incidents. In considering those steps, they looked at what information the operator that led to the error missed, how the error was detected, and how the error was recovered from. From this information, Naikar (2006b) suggested that training protocols could be designed to help operators avoid such errors or recover from those errors when they do occur; however, she did not explicitly discuss how the training would be related to the SRM behaviours that drive the completion of tasks described by the decision ladder.

Finally, Ashoori (2012) gave some preliminary consideration to training at the SOCA phase. Most complex systems involve multiple operators and a division of work is necessary through a process such as a SOCA. Ashoori (2012), in her analysis of CWA in teamwork, noted the necessity of considering training requirements after tasks are divided among team members. Ashoori (2012) and Ashoori and Burns (2012) suggested considering SOCA not only as a distinct phase, but as a revisiting of all phases of a CWA; therefore, training requirements derived at any point during a CWA may be affected by the division of work among various operators and tools once the SOCA phase is considered.

In summary, Lintern and Naikar (1998), Naikar and Sanderson (1999), Naikar and Saunders (2003), Naikar (2006b), Naikar (2009), McIlroy and Stanton (2012), and Ashoori (2012), have all considered training from a CWA perspective, but they have focused on the WDA, CTA, and SOCA phases. This type of research is quite useful as it can be used to define the training requirements of systems in terms of work domain requirements and constraints and teamwork. However, to gain a complete picture of how training for systems described by CWA might be done, consideration of the cognitive aspects of operators in the systems, as described by the WCA phase is needed. Therefore, because the dominant method for WCA is the SRM taxonomy, work on identifying learning needs with respect to SRM is needed. Additionally, once behaviours needed for training are identified, a method of training those behaviours will be needed.

2.5 Learning from a SRM Perspective

It has been discussed that learning of behaviours as described by SRM could contribute to methods of training in CWA-analysed systems. However, as will be shown, the idea of training from a SRM perspective is not a simple one because there is no single way for behaviours to be learned, particularly because of the complexity of ways in which behaviours can be combined to allow an operator to act both correctly and efficiently, as the situation demands. However, regardless of a particular learning path, to learn any SRM-described behaviour, an operator will need to learn to interpret cues properly as signals, signs, and symbols, and he or she will need to learn what actions to take for each cue (Rasmussen, 1983, 1986; Vicente, 1999).

As discussed previously, the cognitive demands of the behaviours decrease from model- to rule- to skill- based behaviour (Rasmussen, 1983, 1986; Vicente, 1999). Therefore, it seems at first glance that it would be desirable to train a person to complete a task using as much skill-based behaviour as possible. Rasmussen, early in the research of SRM, suggested a progression through the behaviours to more rule- and skill- based behaviour from model- based behaviour. Rasmussen (1979; as cited in Sanderson & Harwood, 1988) stated: "The control of human activity shifts from level 3 [model-based] through 2 [rule-based] to 1 [skill-based] in the case of self-instruction [and] from 2 to 1 when an instructor or written procedures are active". In that quote, Rasmussen is suggesting two possible paths through the behaviours as learning occurs. The first path is one starting from reasoning using a model-based behaviour, moving through rule-based as the person develops the knowledge to behave in a rule-based manner, and finally using skill-based as that knowledge is automated. The second path occurs when a person is given procedural instructions on how to complete a task. In that second path, the person consciously executes the instructions until they are well practiced enough such that they become automated (Rasmussen, 1986). These learning paths make sense, and, as Vicente (1999) points out, probably account for some learning, but they don't give a complete picture.

There are more possible learning paths through the Skills-Rules-Models framework. For example, a person can struggle, in a trial-and-error fashion, until he or she can complete a skill at the skill-based level (Olsen & Rasmussen, 1989; Vicente, 1999). For example, as Vicente (1999) pointed out, children learn to walk, a sensory-motor learning task, through trial and error, not by first reasoning about how to walk. A person might also be given a rule to follow, and never practice the rule well enough to complete a task using skill-based behaviour, resulting in a learning path entirely at the rule-based level (Rasmussen, 1986). Furthermore, because skill-based behaviour requires cues presented as signals (Rasmussen, 1983), it would also be possible for a skill not to be developed when the cues to perceive the needed information as a signal do not exist.

There are other possible learning paths through SRM, but as can be seen from the above example, learning is not just a simple progression through the behaviours. In order to explain some of the richness of learning as related to SRM, Olsen and Rasmussen (1989) and Vicente (1999) used Dreyfus and Dreyfus's (1980, 1988) five stages of learning, which are novice, advanced novice, competent performer, proficient performer, and expert. (The names of the stages vary between various publications.) Olsen and Rasmussen (1989) noted that the learning process is really a continuum, but also that the five stages are a useful distinction for considering the support of learning. The five stages, as defined by Dreyfus and Dreyfus (1980, 1988) are the following:

1. *Novice:* A novice starts by observing components of a system without consideration of a larger picture of the system. Similarly, a novice acts within the system using context-free rules or very effortful reasoning.
2. *Advanced Novice:* After gaining some practical experience, an advanced novice begins to get a sense of how information and behaviours they are developing fit into the larger picture of the system. An operator might not be explicitly taught what the relevant context information is, but will still start to recognize familiar situations.
3. *Competent Performer:* As an operator learns more context-related information, the competent performer will start to get an intuitive sense of the context and what information and rules are important. In some situations, a competent performer can run into difficulty, as he or she may try to consider too much previously ignored contextual information without being able to prioritize. However, eventually, an operator learns what is important to consider.
4. *Proficient Performer:* A proficient performer has a more complete sense of the context in which information is perceived and knowledge applied. He or she can prioritize information and actions. The operator also has a developing sense of normal situations and actions in a system.
5. *Expert:* An expert's behaviour is based largely on previous experience, and recognizing situations as familiar. The expert knows what normally works, but will recognize when a situation varies from normal and, therefore, when more effortful consideration of action is needed.

Olsen and Rasmussen (1989) examined the relation between SRM and Dreyfus and Dreyfus' (1980, 1988) stages of learning. Olsen and Rasmussen (1989) noted that an expert is likely to be commonly able to rely on rule- and skill- based behaviour to complete tasks efficiently, but they also noted that as the knowledge needed for rule- and skill- based behaviour is developed, the knowledge needed for model-based behaviour can deteriorate. A lack of the knowledge of a system model can make it difficult for an experienced operator to understand the reasons why he or she takes certain actions, even when the actions are the correct ones to take (Olsen & Rasmussen, 1989). In contrast, Olsen and Rasmussen (1989) defined a *reflective expert* as an operator who can operate efficiently using rule- and skill- based behaviour but also maintain knowledge of the model describing a system. Such reflective experts can then recognize when they can no longer perform in a situation using rule- and skill- based behaviour due to unfamiliarity with a situation and, therefore, must revert to model-based behaviour.

On the other end of Dreyfus and Dreyfus' (1988) set of stages, Olsen and Rasmussen (1989) also attached another stage known as the pre-novice. The pre-novice stage recognizes that an operator new to a domain does not enter training as a blank slate. A new operator has existing knowledge and preconceptions that shape how he or she approaches the system, the work domain, and learning (Olsen & Rasmussen, 1989). As a result of the pre-existing knowledge, a pre-novice, and then a novice, is not restricted to blindly applying context-free rules, but can engage in some reasoning in a work domain, though, as is shown later, when discussing cognitive load theory, this reasoning is likely to be quite effortful.

The work of Dreyfus and Dreyfus (1980,1989) and Olsen and Rasmussen (1989), and the previously discussed work on the mixing of behaviours, shows that the learning of behaviours to act efficiently in a system is not a simple matter of learning some rules and practicing them until a skill is developed. As discussed, it is necessary for an operator to be able to use each of model-, rule-, and skill- based behaviours in appropriate combinations to complete tasks correctly and efficiently. Therefore, because switching between behaviour types might be appropriate depending on the circumstance, and knowing when to switch is important, the use of appropriate levels of behaviour is a marker of a high level of expertise in task completion (Olsen & Rasmussen, 1989).

It has been shown that the idea of a progression of behaviour from model- to rule- to skill- based is an incomplete one; however, the idea of progression is not completely lost. As an operator attempts to learn rule- or skill- based behaviour, their actions and learning will need to be controlled by some method. Rasmussen (1986) described how model-based behaviour might be used during the development of the knowledge for rule-based behaviour. In such a case, the rules are developed empirically, as an operator solves problems (Rasmussen, 1986), a process Olsen and Rasmussen liken to Anderson's (1983) compilation of knowledge. Likewise, rule-based behaviour might be used to control behaviour during the development of skill-based behaviour, through practice, similar to Anderson's (1983) strengthening (Rasmussen, 1986). Rasmussen (1983) also described the possibility of rules being communicated explicitly from others, but these rules still need to be remembered and practiced.

Methods are needed to teach new operators the variety of knowledge needed to allow them to engage in the various skill-, rule-, and model- based behaviour to operate a system correctly and efficiently. The knowledge that operators will need includes the knowledge needed to interpret cues correctly as symbols, signs, or signals, and know how to act appropriately given the various cues. In keeping with the philosophy of CWA, a teaching method must allow the design of instructions that conveys information for new systems and unexpected situations. In other words, the analysis of what to teach must be formative in nature and the knowledge taught must be flexible.

2.5.1 4C/ID Training Design Method

The above work describes in a very general sense how domain knowledge can be used and trained; however, how instructional design and learning can be put into practice needs to be considered. Van Merriënboer (1997), van Merriënboer, Clark, and Croock (2002), and van Merriënboer, Kirschner, & Kester (2003) have described the Four-Component Instructional Design method (4C/ID) for designing instruction for complex systems. This method is not widely present in the human factors literature (though there is some discussion in Darken, 2009 and more extensive discussion in Pass and van Gog, 2009), but the method does describe an extensive method for training and has been related back to the SRM taxonomy (in van Merriënboer, 1997).

The 4C/ID method is intended for training in very complex systems where the expected training time is at least 100 hours, and possibly measured in months or years (van Merriënboer, 1997; van Merriënboer, Clark, & Croock, 2002). The method, founded on Cognitive Load Theory (see Sweller, 1988), is based on the recognition that the learning of many complex behaviours cannot be done in pieces, but must be done in an integrated fashion. The alternative of practicing a task in pieces can be ineffective in some instances because essential features of the task can be lost when it is broken into pieces (van Merriënboer et al., 2002). Van Merriënboer (1997) and van Merriënboer et al. (2002) still claimed that there can be a division of complex behaviours into constituent behaviours, but that for training to be effective when constituent behaviours must be executed synchronously, an operator must practice the behaviours synchronously.

Van Merriënboer (1997) and van Merriënboer, et al. (2002) divided constituent behaviours into recurrent and non-recurrent constituent behaviours. Recurrent constituent behaviours are ones that have similar cues and exit conditions from use to use and the cues and exit conditions for non-recurrent behaviours vary more widely from use to use (van Merriënboer, Clark, and Croock, 2002).

Van Merriënboer, Clark, and Croock (2002) claimed that the type of training needed for the different types of behaviours varies. They suggested that recurrent behaviours can be trained using intense, routine practice until automatic, while non-recurrent behaviours must be trained as a part of a whole-task training routine where the relevant knowledge can be connected to what the operator already knows. In order to train the various constituent behaviours, van Merriënboer, 1997 and van Merriënboer, Clark, and Croock, 2002, presented four components of their instructional design method: learning tasks, supporting information, part-task practice, and just-in-time (JIT) information, each of which is described in brief next (all from van Merriënboer, Clark, & Croock, 2002):

Learning Tasks are tasks that are intended to be whole tasks in the work domain, albeit possibly executed with some cognitive support, which allow the operator to develop the non-recurrent behaviours needed. As the behaviours start to form, the cognitive supports are gradually reduced.

Supportive Information is the support that is provided to operators as they develop behaviours during the learning tasks. These supports might include textbook-based theories, methods of guided discovery, information about possible cognitive strategies, and feedback. This information is intended to help with the development of accurate mental models.

Part-Task Practice is used to develop recurrent behaviours. In part-task practice, the recurrent behaviours are practiced in relative isolation and can be over trained to the point of automaticity.

JIT Information is information that the operator needs to complete the part-task practice. The information should be presented in small chunks, to avoid the need for memorization. The information could include demonstrations and feedback.

All of the above components are combined in a sequence appropriate to the behaviours that the operator needs to act in a system (van Merriënboer, 1997; van Merriënboer, Clark, & Croock, 2002). The types of training can be related back to SRM. As van Merriënboer (1997) pointed out, a goal of the 4C/ID model is to develop the knowledge needed to use efficient rule-based behaviour in a system. He also claimed that development of reflective expertise was a goal of 4C/ID as it allowed operators to use existing knowledge to act in new scenarios, through a process of transfer, a concept further discussed later.

The 4C/ID training is a potentially useful method that has been considered as a basis of instructional design in a number of domains such as computer training (Kehoe et al., 2009), engineering drawing (Guochen, Kuishan, & Jing, 2012), geographic field work (Pérez-Sanagustín, Santos, Hernández-Leo, & Blat, 2012), and medical training (Cook, Beckman, Thomas, & Thompson, 2008; Tjiam, Schout, Hendrikx, Scherpbier, Witjes, and van Merriënboer, 2012). However, as is shown in the next section, there are some scenarios were the 4C/ID model might not be an ideal method of training, leaving some room, while drawing on some of the concepts in 4C/ID, for a different method of training for building SRM-described behaviours.

2.5.2 Needed work on Training Related to SRM

Previously, in section 2.4, the important of training as related to SRM-behaviours was noted to fill in a gap in the training literature related to CWA. Furthermore, Dankelman et al. (2004) and Dankelman (2008) noted that SRM is useful in the consideration of training requirements. They described the behaviours needed to perform various medical procedures in terms of SRM; however, they noted that there is not currently sufficient research into the needs and methods of training to develop the behaviours defined by SRM. To fill the gap noted by Dankelman et al. (2004) and Dankelman (2008), consideration of training needs and training methods from a SRM perspective is needed.

A method for the determination of training needs will need to be based on the fact that system operation involves a mixture of behaviour types, particularly in novel scenarios. As discussed, the cognitive demand of a task could be reduced if an operator uses rule- and skill- based behaviour as much as possible, because those types of behaviours impose lower cognitive demands (Vicente, 1999). Therefore, a method of identifying rule- and skill- based behaviours that can be used as part of model-based behaviour in the completion of novel tasks would be useful. Furthermore, because SRM is often used in the context of a CWA, a stronger connection between training of SRM behaviours and the other phases of CWA is needed.

In this dissertation, the focus is on the determination and training of rule-based behaviour. Skill-based behaviour requires more extensive practice and would follow from extended practice of rules (Rasmussen, 1979, as cited in Sanderson and Harwood, 1988); however, there will be room in future research for identification and training of skill-based behaviours.

In addition to identifying training needs as described by SRM, methods of meeting those training needs are needed. The 4C/ID method (van Merriënboer, 1997; van Merriënboer, Clark, & Croock, 2002) provides one potentially useful method of teaching complex skills described by the SRM taxonomy for some complex systems, but for other systems, the complete method may not be suitable, for three main reasons.

First, the 4C/ID method is intended for very lengthy training protocols, where learners will spend hundreds of hours training, making it worthwhile to invest the needed effort to design very extensive and detailed training protocols (van Merriënboer, 1997; van Merriënboer et al, 2002). However, in some cases, CWA has been used for systems that still could involve behaviours as described by SRM (for example Cornelissen, Salmon, Jenkins, and Lenné, 2012), but where the operators will not spend 100s of hours on explicit training protocols. For such systems, simpler methods for designing training protocols are needed.

Second, the division between recurrent and non-recurrent behaviours may not make sense in some systems, particularly when it is unknown how multiple behaviours might be combined or chained together to operate a system in unfamiliar and unexpected situations. In such situations, model-based behaviour would be employed, and as part of that behaviour, various rules-based behaviours might be combined in novel ways (Vicente, 1999). For such systems, it would be difficult to identify the recurrent constituent behaviours that could be trained in an isolated manner using part-task training, making the part-task training component of 4C/ID not particularly useful in such training domains.

A related third point is that in some situations the part-task training proposed by van Merriënboer, (1997) and van Merriënboer et al. (2002) can be ineffective. As Wickens et al. (2013) showed, part-task training can be ineffective when the part-tasks must eventually be performed concurrently, because operators do not practice the time-sharing abilities that will be needed when the operator must complete a whole task. Instead, Wickens et al. (2013), recommend Gopher, Weil, and Siegel's (1989) variable-priority training pattern of training, where whole tasks are practiced, but different parts of the task emphasised. Therefore, in some systems, the use of variable-priority training may be advantageous over the use of part-task training.

Despite the limitations in using 4C/ID, its foundation of training participants using whole task scenarios is important for complex systems (van Merriënboer, Kirschner, and Kester, 2003). As will be seen, when training transfer is discussed, training in whole-task scenarios is particularly important where future operating scenarios are unknown, because, during training, the operators need to see how rule-based behaviours that they develop affect the operation of the system, so they can predict what behaviours are appropriate in a particular context. However, as recognized in 4C/ID, the use of whole-task training can be cognitively overwhelming during training. Therefore, methods of training rules in whole-task scenarios, but with reduced cognitive load, are needed.

Two areas for work in training as related to SRM have been identified: the identification of training needs as described by SRM and the development of methods for teaching the behaviours identified in an analysis of training needs. Both of these issues are examined in this dissertation through either the theoretical work in Chapter 4 or the empirical work in Chapters 5 and 6.

In the next section, training transfer is reviewed and its relationship to training rule-based behaviours is considered, as a step towards the development of methods to identify need rule-based behaviours in completing novel tasks. Then CLT, which is a basis for 4C/ID, is reviewed and be later adapted to train the rule-based behaviours identified as training needs.

2.6 Transfer of Training

In developing a training program based on the SRM taxonomy, it first must be determined what types of behaviours need to be trained for. It was discussed, in the introduction, that some of the scenarios in which a system might be used are unknown at training time (Woods & Roth, 1988; Vicente, 1999); therefore, the behaviours an operator can use when operating a complex system should be usable in new situations. The use of existing knowledge and behaviours in new situations is the essence of training transfer, so ideas from training transfer can be used to consider what behaviours an operator needs to learn to be able to operate a tool in future unknown scenarios.

The question of how people use knowledge gained in one situation in a new one is a complex question that has had many answers over millennia of consideration and study. It seems clear that humans are able to apply knowledge to new situations; otherwise, without the ability to apply knowledge in new situations, formal education would be useless, because students would need to be trained to do a specific job and only that job. Moreover, without the ability to apply knowledge to new problems, humanity would not have made all the technological and scientific discoveries that have been made. However, it is not easy to establish what aspect of human cognition allows humans to apply knowledge in new situations. Some people claim there exists a phenomenon called transfer, which is, roughly speaking, the ability to apply knowledge in new situations (Thorndike, 1906; Haskell, 2001). Others have claimed that transfer does not exist as a distinct phenomenon, and its apparent existence is just an epiphenomenon of other cognitive processes (Detterman, 1993;

Bransford & Schwartz, 1999; Hagar & Hodkinson, 2009), though, in some cases, failure to find transfer could have been a result of the definition of transfer used.

There have been a variety of definitions of transfer that vary in some degree from each other. Detterman (1993) defined transfer as "the degree to which behaviour will be repeated in a new situation" (p.4). If this definition is used, the transferred behaviour must be repeated exactly as it was performed before, a pretty narrow definition, as there is no room for behaviour to be adapted to any slight differences between the original and new situation.

Haskell (2001) gave a more flexible definition: "Transfer of learning is our use of past learning when learning something new and the application of that learning to both similar and new situations." (p.xiii). This definition allows flexibility in that previous learning can be applied in a new situation. Royer, Mestre, and Dufrense (2005), give what is possibly an even broader definition: "transfer is a term that describes a situation where information learned at one point in time influences performance on information encountered at a later point of time" (p. vii). With exact repetition not required, Haskell (2001) and Royer et al. (2005) give broader definitions of transfer, but blur the line between transfer and learning. Going further, some authors (i.e. Hager & Hodkinson, 2009, Bransford & Schwartz, 1999 and Schwartz, Bransford, & Sears, 2005) question if transfer even exists as a phenomenon distinct from learning, as will be discussed.

How similar or not a situation needs to be to another for transfer to be said to occur is a matter for debate, and often different types of transfer are said to occur for different levels of similarity (Haskell, 2001). Haskell (2001) defined six different types of transfer that occur depending on the similarity of the situations. Haskell noted that even with the different types of transfer defined, similarity is often subjective and there is no clear boundary between the different types of transfer, though it seems that it is more likely that transfer could occur the more similar two situations are.

In the remainder of this section, to examine how transfer can be conceptualized in a modern context, some current research in training transfer is presented, preceded and framed by a review of some of the historical views on transfer.

2.6.1 Doctrine of Formal Discipline

Before the explicit consideration and testing of the ideas of transfer, the prevailing view of human learning was the *doctrine of formal discipline* (Haskell, 2001). Advocates of the doctrine of formal discipline viewed the mind as a muscle in the sense that any type of exercise strengthened the mind and made it more adept at any number of tasks (Haskell, 2001). For example, one could posit that learning Latin trained the mind to be careful and disciplined in thought, and that such discipline would carry over to the study of mathematics, where care and discipline is also needed. This doctrine can be traced back as far as Plato, who wrote in The Republic (1977 trans.) “And have you further observed, that those who have a natural talent for calculation are generally quick at every other kind of knowledge; and even the dull if they have had an arithmetical training, although they may derive no other advantage from it, always become much quicker than they would otherwise have been.”

2.6.2 Identical Elements Theory

The doctrine of formal discipline persisted for millennia, and continues to do so in some forms (Haskell, 2001). However, Thorndike (1906) questioned the doctrine of formal discipline and studied students as they were taught various subjects and saw that improvement of performance in one subject did not necessarily correlate with improvement of performance in another. For example, he found that improvement in reading did not necessarily correlate with improvement in addition skills. He even found that improvement in neatness of students’ arithmetic papers did not result in improvement in neatness of the students’ spelling papers. Thorndike concluded that it could not be assumed that improvement in one subject corresponded to some general improvement of the mind. He claimed that, at the very least, any theorized coupling of subject skills must be tested.

Thorndike (1906) instead proposed the *identical elements theory* that states “one mental function or activity improves others in so far as and because they are in part identical with it, because it contains elements common to them.” (p. 243) For example, he claimed learning English helps a person in general communications, as writing and speaking well are important to any communication, but learning to write a clear letter to a friend may not necessarily help as much as one would hope with writing an effective business proposal. However, there is a problem with the identical elements theory, as Haskell (2001) and others have pointed out, in that there are rarely tasks that are the same as another, and it is not clear from Thorndike how similar two situations need to be to trigger transfer of knowledge.

Thorndike was trying to determine how learning from one knowledge domain can be reused in another domain given suitable similarity between the two domains. Detterman (1993) questioned if transfer of this type exists, as he noted that many laboratory studies have failed to show transfer. He pointed out several studies where students failed to transfer a solution from one problem to another very similar problem. Detterman noted that in several studies where transfer is claimed to have occurred, for example he cited Judd (1908) and Gick and Holyoak (1980) (both as cited in Detterman, 1993), that the participants had to be explicitly told to use certain analogical problems in order to induce transfer, and he wondered if the effect of someone being told to apply old knowledge in a new situation could be called transfer. Therefore, Detterman questioned if the seemingly everyday transfer that occurs is actually a distinct facet of human cognition appropriately labelled transfer or an epiphenomenon of another cognitive process.

2.6.3 Knowledge Bases

Haskell's (2001) definition of transfer hints at transfer being closely related to learning and starts to move the focus away from a distinct process of knowledge reuse. Continuing with this shift in thought, several authors see learning and transfer as essentially the same thing.

Bransford and Schwartz (1999) and Schwartz et al. (2005) saw transfer not as an end where a person can apply a body of knowledge in new situations, but as a *preparation for future learning*. As part of the idea of application as learning, these authors noted that knowledge, even if not directly applied to future tasks, can change how people interpret tasks and information from the environment and help with the learning or discovery of new solutions. This idea that existing knowledge helps with further learning can be related to Olsen and Rasmussen's (1989) idea of the pre-novice stage of learning, which is a recognition that an operator new to a domain comes with some existing knowledge that shapes their behaviour as they learn to complete tasks in the domain.

The preparation for future learning idea is consistent with Hagar and Hodkinson (2009) where they described learning as the modification of a relational web of knowledge. They claimed that any cognitive action, be it acquiring new knowledge or applying knowledge to a problem, is a learning process and results in a change in the relational web. They saw learning as a continuous, on-going process. Even during what is traditionally thought of as evaluation processes, tests and such, they see learning occurring. Because learning is on-going, an important purpose of learning and testing is to ensure that an operator is on a trajectory for future learning and not just to ensure he or she has mastery of a set of knowledge (Bransford and Schwartz, 1999).

If preparation for future learning is the broad goal of instruction, the question becomes what specifically to teach as part of instructional processes that will be of use during the later application-learning process. Haskell (2001) saw a large knowledge base of domain-relevant information as an essential prerequisite of the ability to handle new situations. He claimed the knowledge base is needed to deal with the many variations that can occur in new situations and that many apparent failures of transfer are in fact due to an insufficient knowledge base. Haskell stated this view within the context of a discussion of transfer. However, even if these new situations are approached through the concept of a continuous learning process, rather than a transfer process per say, a large amount of knowledge will be needed as a person forms a solution to a problem in a new situation. If a person had to generate all the possibly relevant knowledge from scratch, his or her cognitive system would be overwhelmed, or he or she would take too long to produce a solution. Therefore, a large knowledge base that can be drawn on during problem solving in novel situations is needed, and as Singley and Anderson (1989) pointed out, some practice is needed for people to be able to transfer the knowledge.

There are a number of cases where researchers have noted that large knowledge bases of examples are essential to expert behaviour. De Groot (1946/1965) and then Chase and Simon (1973) found that expert chess players can easily reproduce chess board configurations that they saw briefly, where novices cannot create similar reproductions. However, experts can only successfully reproduce a configuration if the configuration results from a real game; they aren't any better than novices at reproducing random configurations. This result suggests the experts draw on a large database of chessboard configurations (Chase & Simon, 1973), presumably because the database helps the players recognize familiar chess configurations and then determine a likely successful move. As another example, Klein (1999) studied how fire fighters make decisions in time-sensitive, high-pressure situations. He found that expert fire fighters have a large knowledge base of previous experience that they used to rapidly determine how to behave and what to expect in a current situation.

Klein's (1999) work on recognition-primed decision making has a large perceptual component to it, where the decision makers recognize familiar cues that indicate a situation's similarity to previous situations (Vicente and Rasmussen, 1992). Day and Goldstone (2012) similarly reinterpreted the transfer phenomenon from a perceptual perspective, claiming that recognition of perceptual features of a situation may trigger an underlying mental model and may play a crucial role in the facilitation of transfer.

2.6.3.1 Knowledge Base For Rule-Based Behaviour

From a SRM perspective, when an operator is using a tool in a new situation, he or she will need to engage in cognitively intensive model-based behaviour in order to discover novel solutions (Vicente, 1999). However, in some such situations, an operator might be able to use less cognitively intensive rule- based behaviour as part of their model-based behaviour (Vicente, 1999). Therefore, having knowledge that allows the use of model-based behaviour to discover novel solutions will be important, but also of importance will be the ability to recognize some familiar aspects of new situations where rule-based behaviour previously learned can be re-used. The ability to recognize familiar situations will depend on the perceptual recognition of signs, tying the transfer of rule-based behaviour into the interpretation of transfer by Day and Goldstone (2012).

Furthermore, during new situations, operators should be in a position to learn new knowledge from information in the environment that will allow more efficient rule- and skill- based behaviour in the future when similar situations are re-encountered. As part of this learning process, operators will have to learn to interpret familiar cues that might appear as part of the new situation, when re-encountered, as the signs that can be used to determine what rule-based behaviours will be usable as part of future model-based behaviour. In other words, an operator should be building their knowledge base, allowing future rule-based behaviours, as he or she completes tasks.

2.6.3.2 Avoiding Errors and Use of Context

A corollary of preparing operators to act in new scenarios is that the operators will also need to know when the various rule-based behaviours are appropriate to use, and even more importantly when those behaviours are not appropriate to use (Reason, 1990). Reason (1990) enumerates a number of ways in which rule-based behaviour can be misapplied in inappropriate situations as a result of failing to distinguish the cues indicating one situation from another. Therefore, another requirement of successful transfer is having appropriate knowledge to avoid the misapplication of known behaviours in inappropriate situations (Perkins and Salomon, 2012; Reason, 1990). In particular, operators need to recognize both signs, which indicate that a rule applies, and countersigns, which indicate that a rule does not apply (Reason, 1990). Relatedly, as part of their preparation for future learning concept, Bransford and Schwartz (1999) and Schwartz et al. (2005) viewed transfer as not just applying knowledge but being able to compare and contrasts situations and know what knowledge to apply.

2.6.3.3 Training for Transfer

When considering teaching for problem solving, both Sweller and Cooper (1985) and Reed (1993) claimed that the use of multiple examples of a particular problem type is beneficial. Multiple examples allow students to see problem solving patterns, including what sub goals are useful (Reed, 1993). Reed (1993), in the context of a discussion on the best way to teach students knowledge that they can transfer to problems of a similar nature, provided evidence that using examples, reinforced by explicit, domain relevant procedures, is a better method than the sole use of general knowledge for promoting successful transfer. Similarly, Day and Goldstone (2012) noted, some context and examples are needed, possibly complemented with assistance in detecting appropriate abstractions and generalizations, in teaching concepts in order to promote understanding of the appropriate context for the use of knowledge. Such a need for context is directly related to van Merriënboer's (1997) claim that whole-tasks are needed for training so that operators understand when certain constituent behaviours are appropriate.

However, Day and Goldstone (2012) also pointed out that the examples must be varied to promote the formation of knowledge general enough to be useful in future situations. The need for examples and generalizations can be related back to the idea that knowledge of a system can exist for different levels of generalization as described in a work-domain abstraction hierarchy. More general concepts will be found at higher levels of the abstraction hierarchy and give the purpose of the lower level concepts and objects (Rasmussen, 1985, 1986). Furthermore, Bransford and Schwartz (1999) noted that contrasting contextual examples are also important, so that operators can learn to differentiate appropriate contexts for the application of some knowledge from inappropriate contexts.

Despite the importance of context during training, contextual material needs to be design such that it does not overwhelm an operators' cognitive resources (Day and Goldstone, 2012). To examine how contextual material can cognitively overwhelm an operator during training, and develop methods of reducing the interference of contextual material, CLT, which examines how the design of learning material can affect learning outcomes, can be used.

2.7 Cognitive Load Theory

In any kind of training, including the training of rule-based behaviour, the limits of human cognition must be considered to ensure the training is effective. As has been known since Miller (1956), human working memory is extremely limited. Miller estimated the number of items that a person can process in working memory to be about seven. Cowan (2001) found the number to be even smaller, especially when the items are being actively processed in working memory. Regardless of the exact limit, the number is small and, therefore, working memory resources must be considered when designing instructional material, or a student's working memory could be overwhelmed, resulting in little or no learning (Sweller, 1988). Sweller (1988) introduced *Cognitive Load Theory* (CLT) to describe how working memory resources are used while a student is learning, particularly when the student is learning through traditional problem solving, where a student is given problem start and end states, along with a set of valid transformations, and no other assistance.

CLT is based on the premise that learning involves the development of schemas. Sweller (1988) noted Simon and Simon's (1978) finding that domain novices and experts approach problem solving differently. Novices tend to solve problems using a means-ends strategy: they start from both the problem start and end states and try to generate steps that will solve the problem. Experts are often able to classify problems by solution type and work in a forward-only sequence from problem start state to end state. Sweller (1988) claimed that an expert in a domain has an extensive library of schemas in his or her long-term memory that allows the categorization of problem by solution type. Therefore, the process of becoming an expert in a domain is the process of storing large numbers of schemas in long-term memory.

2.7.1 Schemas

Schemas are an important concept in CLT, but different authors have used different definitions over a long history of the use of the term. Thorndyke (1984) noted that the concept of schemas goes back at least to 18th century German philosopher Immanuel Kant. Sweller (2011) noted the concept became popular in psychological research in the work of Piaget (1928 as cited in Sweller, 2011) and Bartlett (1932). Bartlett (1932) defined a schema as "an active organization of past reactions, or of past experiences, which must always be supposed to be in operation in any well-adapted organic response" (p. 201).

Thorndyke (1984) gave a related definition and outlined some properties of schemas that allow the use of schemas as an abstraction of how humans store information. Thorndyke defined a schema as "a cluster of knowledge representing a particular generic procedure, object, percept, event, sequence of events, or social situation" (p. 167). Thorndyke saw schemas as templates of various levels of abstraction that details can be filled into. As a person engages a particular cognitive process, he or she matches the schemas in his or her long-term memory with the situation being experienced. The schema then allows the person to identify a situation and know how to act or respond.

According to Thorndyke (1984), schemas can be formed as people have various experiences. A person who figures out how to solve a particular problem might form a schema to help him or her solve that same problem if encountered again. However, as is discussed in the next section, a person might solve a problem, but still fail to form a schema that will be of use in the future. I will use Thorndyke's definition and properties of schemas. His definition and the properties he describes help with reasoning about the role of schemas in learning and problem solving.

There is a close relationship between schemas and rule-based behaviours. Schemas allow an operator to recognize a situation at various levels of abstraction and know how to act, and knowing how to act in a situation is the characteristic of rule-based behaviour (Rasmussen, 1986). So, it can be proposed that the development of appropriate schemas should allow for rule-based behaviour.

Schemas are quite useful in reducing the cognitive demand of a task, but schema formation is an effortful process that consumes limited working memory resources (Sweller et al., 2011). Consideration of CLT leads to methods that promote formation of useful schemas, which can in turn reduce cognitive load when a person encounters a similar situation in the future.

2.7.2 Intrinsic and Extraneous Cognitive Load

The current version of CLT as described in Sweller, Ayres, and Kalyuga (2011) defines two types of cognitive load: intrinsic and extraneous. These types of cognitive load help describe the cognitive demands placed on a person's working memory by a piece of information. After discussing the two types of cognitive load, the types cognitive resources that can be directed to processing the cognitive load and learning the information will be discussed.

Both types of cognitive load, intrinsic and extraneous, are defined in terms of element interactivity (Sweller, 2010; Sweller et al., 2011). To understand element interactivity, an understanding of the term element is needed. There is no way to determine what constitutes an element without considering a person's knowledge. Sweller et al. (2011) pointed out that what constitutes an element is not universal and depends on a person's existing knowledge. They gave the example of how different people might process the word "cat". A child learning to read might see "cat" as a sequence of three elements, three characters, to be sounded out. A literate person would see "cat" as one element, the word as a whole. A person unfamiliar with the Latin alphabet might see "cat" as a meaningless set of squiggles.

Element interactivity is a measure of the number of elements of a piece of information that needs to be processed simultaneously in order to achieve an understanding of the information. Sweller et al. (2011) defined intrinsic and extraneous cognitive load in terms of element interactivity. Element interactivity essential to the task-relevant information being processed imposes an intrinsic cognitive load, while element interactivity not essential to the task-relevant information being processed imposes an extraneous cognitive load.

Intrinsic cognitive load is a measure of the complexity of material to be learned (Sweller, 2010) and is a result of interacting elements of the training material relevant to a learning goal (Sweller, 2010; Sweller et al., 2011). Intrinsic cognitive load is a property of the instructional material, but, because element interactivity of a piece of instructional material can vary from person to person, the perceived intrinsic cognitive load of a piece of instructional material can vary.

Not all element interactivity contributes to intrinsic cognitive load. Sweller (2010) and Sweller et al. (2011) defined *extraneous cognitive load* as the load imposed by element interactivity present in the material but that is not essential to what the person is trying to learn. For example, when trying to teach a student to add two numbers, instructions presenting the task as a word problem would increase cognitive load of the instructional material, and that additional cognitive load would be extraneous to the task of learning to add two numbers. As with intrinsic cognitive load, the amount of extraneous cognitive load that a piece of training material imposes on a trainee will depend on how he or she perceives the amount of element interactivity, which, as discussed above, depends on a trainee's knowledge level (Sweller et al., 2011).

It should be made clear that a large amount of information does not necessarily constitute extraneous cognitive load, or even high intrinsic cognitive load. Intrinsic cognitive load was defined in terms of element interactivity in Sweller (1994) and Sweller and Chandler (1994), but, as noted in Sweller (2010), up until the 2010 publication, there had been little effort to assess the cause of and clearly define extraneous cognitive load. In Sweller (2010) and Sweller et al (2011), CLT was modified to define extraneous cognitive load in terms of element interactivity, making element interactivity more central to the theory. This change emphasized the point made in Sweller and Chandler (1994) and Schnotz and Kürschner (2007) that CLT and the idea of cognitive load are focused on the number of elements a person needs to hold simultaneously in their working memory to achieve understanding of the material. Both Sweller and Chandler (1994) and Schnotz and Kürschner (2007) compared two tasks often involved in learning a language: memorizing vocabulary and comprehending sentence structure, to clarify what kinds of tasks impose cognitive load. Even though memorizing a long list of vocabulary is a difficult task, it is a task that should impose low cognitive load, since only one word, or word pair, needs to be held in working memory at a time. On the other hand, understanding a sentence should require the simultaneous processing of many elements in working memory, imposing a higher cognitive load.

Some attempts have been made to measure the amount of cognitive load imposed by a piece of instructional material. In order to generate an a priori estimate of element interactivity of a piece of instructional material, Sweller and Chandler (1994) performed a task analysis and isolated all the considerations that they believe a student might need to process in performing a number of tasks. The authors noted that the result of such an analysis is surely only an estimate, because the true element interactivity will depend on the student's existing schemas. However, Sweller and Chandler (1994) used these estimates to make a hypothesis about the relative cognitive load imposed by two sets of instructional material. Experimentally, they showed that the students who experienced lower estimated cognitive load performed better on high-interactivity learning tasks.

In addition to using a priori estimates of cognitive load, post-hoc attempts have been made to measure the cognitive load placed on students during learning (e.g. Ayres, 2006b; DeLeeuw & Mayer, 2008; Wirth, Künsting, & Leutner, 2009; Gerjets, Scheiter, & Catrambone, 2004, 2006). Of particular relevance to this dissertation is the work of Gerjets et al. (2004, 2006) that showed that the NASA-TLX (Staveland & Hart, 1988), a method of measuring workload frequently used in human factors studies (Hart, 2006), is responsive to manipulations in cognitive load (see also Sweller et al., 2011).

2.7.3 Cognitive Resources

Sweller et al. (2011) claimed that intrinsic and extraneous cognitive load are additive, and a piece of material imposing both types of cognitive load must be processed using a student's limited cognitive resources as he or she strives to understand the material. The cognitive resources that are used in processing the cognitive load of a piece of material can be divided into germane and extraneous cognitive resources (Sweller et al., 2011).

Sweller et al. (2011) defined *germane cognitive resources* as those resources directed towards processing the elements contributing to intrinsic cognitive load, which are the elements germane to the learning goal. As information related to intrinsic cognitive load is processed, new schemas will form, eventually reducing the intrinsic cognitive load imposed by the information.

Extraneous cognitive load also needs to be processed by a student's limited cognitive resources. Such processing, by *extraneous cognitive resources*, also forms new schemas, but, by definition, these new schemas are not relevant to the chosen learning goal (Sweller et al., 2011). If too many cognitive resources are focused on processing extraneous cognitive load, a student is unlikely to form schemas relevant to the learning goals. Even if all the load of the learning material is intrinsic, if the student is overwhelmed, he or she will be unable to process the material, so useful schemas will not be formed (Sweller, 2011).

2.7.4 Cognitive Load Theory-based Teaching Strategies

To avoid overwhelming cognitive resources during problem solving and learning, novices need guidance. Providing too little or no guidance can result in students getting confused while problem solving and learning little (Kirschner, Sweller, and Clark, 2006) or failing to notice essential features of the problem and solution (Mayer, 2004). Sweller (1988) and Tarmizi and Sweller (1988) pointed out that problem solving can be a poor method of teaching, particularly when the students must resort to a means-ends method. If a student is left to solve a too-difficult problem with little guidance, his or her working memory resources will be focused on solving the problem, and no resources will be available for formation of schemas relevant to the goal of understanding the problem (Sweller, 1988). Therefore, even if a student does eventually solve a problem with minimal guidance, he or she may not remember how he or she solved it or not have noticed what important steps and information was in the solution.

Sweller and Cooper (1985) suggested worked examples as a better alternative to problem solving and as a method for encouraging schema formation. Worked examples allow students to focus cognitive resources on the formation of task-germane schemas and not on problem solving (Sweller et al., 2011). A properly chosen worked example focuses on schemas that are essential to efficiently solving the type of problem being presented (Sweller et al., 2011).

Van Merriënboer and Krammer (1987) and van Merriënboer (1990) investigated completion problems, which are similar to worked examples, as a tool for instruction. In completion problems, students are either given a nearly solved problem and asked to complete the problem or are given a solution to one problem and asked to modify the solution to solve a closely related problem. van

Merriënboer (1990) noted that completion problems allow a student to focus on one particular task without having to expend cognitive resources on other tasks that are part of a larger problem. Completion problems have an advantage over worked examples in that with worked examples a student might only give cursory consideration to and fail to effectively learn from the example, whereas completion problems force the student to consider how the partial solution relates to part of the problem he or she must complete, forcing processing of the problem in working memory causing the formation of schemas (van Merriënboer, Kirchener, & Kester, 2003).

Another method of reducing cognitive load is pre-training. Clarke, Ayres, and Sweller (2005), Schnotz and Kürschner (2007) and Sweller et al. (2011) claimed that by pre-training someone to complete part of a task or use a tool to complete a task can reduce cognitive load in the future when the whole task is completed. Ayers (2006) found that isolating parts of a mathematical learning task can be of benefit for students with low ability, and students of all levels of ability benefited from an approach that gradually moved from part training to whole-problem training. Clarke et al. (2005) gave an example that is particularly related to the application discussed later in this dissertation. Clarke et al. (2005) studied participants' learning of spreadsheet software and mathematics. The authors found that participants in the study who had low knowledge of the spreadsheet software and learned the software and the mathematics sequentially performed better on tests of the mathematics than participants who learned both topics concurrently. The authors also measured the student's cognitive load during instruction via a self-rating scale of difficulty from "extremely easy" to "extremely difficult", but they found no significant main effect of learning the software and the mathematics concurrently on cognitive load; however, they did find that the participants with high spread sheet experience experienced less cognitive load in the concurrent condition. In the conclusion to their paper, Clarke et al. (2005) recommended, based on the improved learning by the sequential group, the learning of the software before the domain material. They suggested that only once the software is sufficiently learned should it be used to assist with learning in other domains.

The work of Clarke et al. (2005) is consistent with the theoretical work of Schnotz and Kürschner (2007) and Sweller et al. (2011), in which they claim the intrinsic cognitive load imposed by a task, or part of a task, can be reduced by pre-training part of a task to induce the formation of schemas that then reduce the intrinsic cognitive load imposed by the whole task. However, the use of pre-training must be balanced with the previously noted claims of van Merriënboer (1997) and Wickens et al. (2013) that part-task training can be ineffective if important components of the whole task are lost, such as the need for time-sharing of cognitive resources. Wickens et al. (2013) instead recommended the use of training that uses whole tasks, but variably emphasizes different parts of the task.

As a student learns, the interventions discussed can become less useful to the student (Sweller et al., 2011). Renkl, Freiburg, Atkinson, and Maier (2000) and Sweller et al. (2011) noted that as expertise increases, the usefulness of worked examples and completion problems can decrease, an effect known as the *guidance fading effect*. Because of this effect, van Merriënboer, Clark, and Croock (2002) implemented gradually reduced cognitive support in their 4C/ID method. They used a concept called scaffolding, where by cognitive support is given, perhaps by having a student focus only on part of a task (e.g. Wickens et al. 2013), or using cognitive apprentice methods be it though personal instructor support (Collins, Brown, & Newman, 1987; Collins, 2006; Järvelä, 1995) or through technology-based support (Chen, 1995), until the student is capable of taking on more of the task with less or no support.

When a person gains sufficient expertise, the extra guidance can even start to be a hindrance. This latter effect is known as the *expertise reversal effect* (Kalyuga, Ayres, Chandler, & Sweller, 2003). Novices, lacking extensive schemas to guide their processing of information during learning and problem solving, may need extra guidance to reduce the cognitive load placed on them. For example, a math text designed to introduce a new topic may contain extensive step-by-step explanations that reduce the cognitive load for a novice. However, as a student gains more expertise, that extra assistance provides unneeded help and eventually may impose unnecessary cognitive load, because the student has to integrate the extra information with what he or she already knows (Kalyuga, Ayres, Chandler, & Sweller, 2003).

2.8 Background Summary

In this chapter, I have examined human factors frameworks and models. CWA was introduced as a method for describing and designing systems. However, once such systems are designed, there is a need to train operators to use the tools in the system. While some work has been done at the ecological end of the CWA phases, more work remains to be done at the cognitive end. To examine training needs at the cognitive end of CWA, theoretical work and training material development related to the SRM taxonomy is needed, particularly with respect to the identification and training of SRM-described behaviours. Ideas from training transfer were identified as a possible source of guidance in the development of training methods for SRM behaviours; however, as was noted, during transfer training, cognitive load can become excessive, so CLT was reviewed as theory that can be drawn on to control cognitive load.

In Chapter 4, to build a training method for the cognitive end of CWA, three guidelines are presented for identifying training requirements based on the SRM Taxonomy using work from the training transfer literature as guidance. CLT will then be used in the design of training material that allows the operators to focus on the learning of the rules developed. However, the introduction of this method could be aided with an example, so in the next chapter, an example domain of a Computer Algebra System is introduced.

Chapter 3
Exploratory Work

In the previous chapter, a need to develop training methods based on SRM behaviours was identified. In introducing such a method, an example application area will be useful, and such an application is introduced in this chapter. An application area that will illustrate the kinds of behaviours identified in Chapter 1 will need to involve the interaction of human operators with technological tools to complete tasks. Furthermore, the tools should be ones where operators can potentially learn rule-based behaviours that will benefit the operator in their use of the tool. One such application area is Computer Algebra Systems (CAS), a tool used by students in the study of mathematics.

CASs are becoming quite common in math classrooms. Lavicza (2010) completed a survey of 1103 mathematicians in the United Kingdom, the United States, and Hungary and found that more than one half of these mathematicians used a CAS in teaching. Furthermore, Buteau, Marshall, Jarvis, and Lavicza (2010) reviewed 204 papers that concerned the use of CASs in teaching. They found that such systems were used for a variety of purposes including experimentation, visualization, and the presentation of real-world examples. Buteau et al. (2010) also found that Maple was the most frequently mentioned CAS in the papers reviewed with 52 papers mentioning Maple. The second most frequently mentioned system was Mathematica with 43 mentions.

Even with the frequent use of CASs in the classroom, problems with their use still remain. Buteau et al. (2010) noted instances of student frustration and failure to achieve learning objectives. Both Lavicza (2007) and Artigue (2002) have noted the considerable time commitment needed to learn to use a CAS. Pierce, Herbert, Giri (2004) looked at the use of the TI-89 system by 43 students. The teacher of that class felt that the system was used most by the stronger students while the teacher "got the sense that the other group felt lost in the maths and the CAS calculator will only help them get further lost" (p.464). Pierce et al. (2004) concluded that there was a substantial overhead in learning a CASs, which must be overcome, before the use of the CAS would be of benefit in future learning. These conclusions are consistent with Clarke et al. (2005) and Sweller et al. (2011) who suggested that a similar tool, a spreadsheet, should be learned before its use, lest the students experience a higher cognitive load, possibly inhibiting learning.

The frequent use of CASs makes the technology a worthwhile one to study. CASs are a form of a tool that can complete some cognitive work and will be used in situations where the exact use of the tool is unknown at training time. Therefore, there is a potential benefit to analysing CASs with CWA.

A CWA analysis of a CAS will involve a significant cognitive training component due to the considerable effort needed to learn a CAS and difficulties that students can encounter in the use of CASs, if they are not properly trained, as pointed out by Pierce et al. (2004). As previously noted, rule-based behaviour requires fewer cognitive resources than model-based behaviour (Vicente, 1999). Therefore, if students can be trained to use rule-based behaviours when using a CAS, the students may experience reduced cognitive load and find the system more useful.

Maple, used for the first three studies in this dissertation, is a CAS created by Maplesoft. In the next section, a potential cause of cognitive overload when students are using Maple to learn new mathematics is identified. This cause of overload will help to identify where rule-based behaviours might be useful. Then, through a field experiment, what Maple knowledge might be useful to teach students is investigated, which will later be used to develop potential rule-based behaviours for using Maple.

3.1 Field Study

In Robinson and Burns (2009), a field study is described during which issues of cognitive overload during use of Maple were investigated. The overall goal of the study was to detect possible causes of difficulties discussed by Pierce et al. (2004) that can make it difficult for students to use a CAS.

3.1.1 Study Method

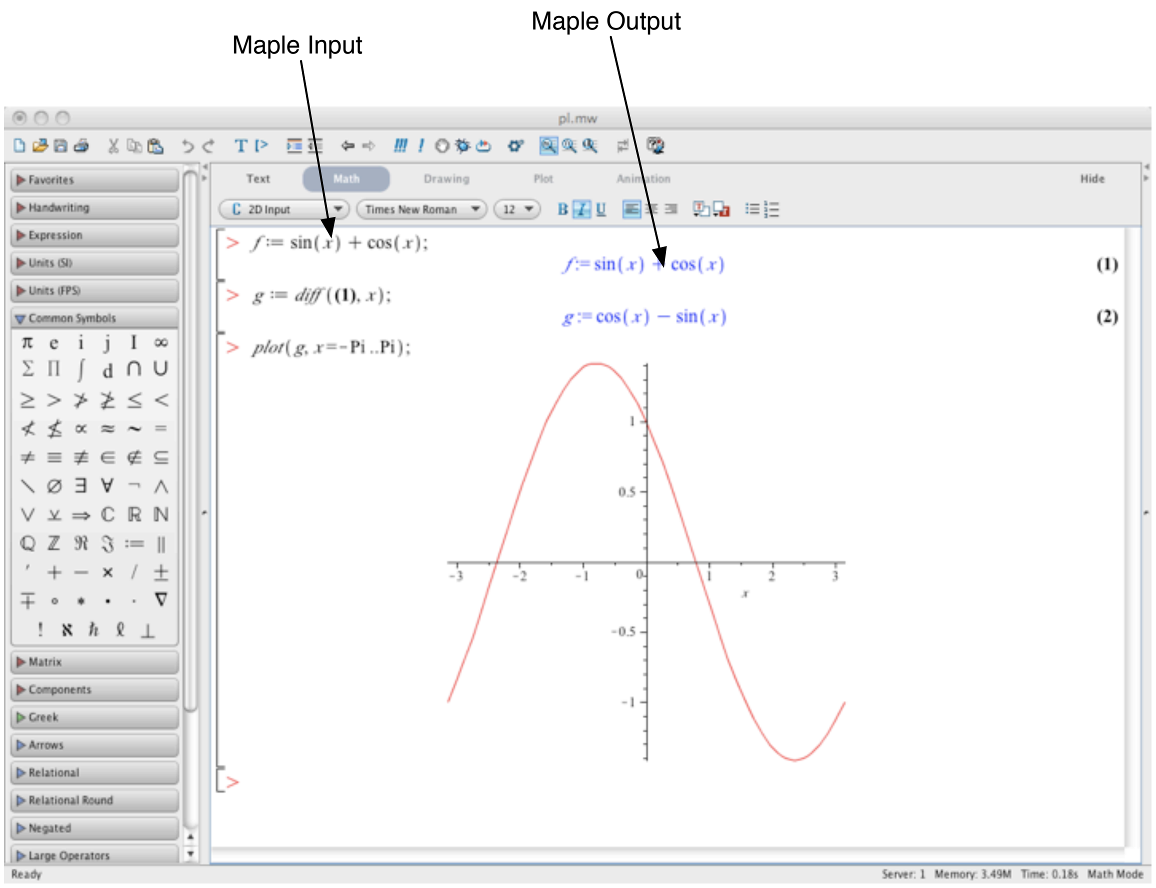
It was noted above that students can benefit from the use of a CAS, such as Maple, but they run a risk of experiencing difficulties due to the cognitive demands the CAS can impose. This study was designed as an observational field study that, as Hutchins (1999), Woods (1993), and Mumaw, Roth, Vicente, and Burns (2000) noted, allows the observation of phenomenon that might be missed in a more structured lab environment and that such studies allow the discovery of new problems to be further investigated. In the interest of identifying some specific problems that students can encounter in a field study manner, an empirical-hermeneutic approach was used (Marmaras & Nathanael, 2005) in making the observations; however, to start to make sense of the meaning of the results, considerations from CLT were used to identify possible sources of high cognitive demand, leading to the combined hermeneutic-positivist approach suggested by Marmaras and Nathanael (2005). Field studies can produce ecologically valid results that are descriptive of a particular work domain; however, there is a lack of control and generalizability in many such studies (McGrath, 1984).

During the study, I observed the behaviour of four students for 90 minutes each as they used Maple to complete an assignment for their first-year calculus course. Also, to better understand the students' thinking when they encountered difficulties (as suggested by Woods, 1993), I questioned and assisted the students when they could not complete a problem.

During the study, notes were recorded on paper about how the students completed the assignments, including what Maple commands they used. Notes were also made of instances where the students had difficulties with the assignment, as indicated by situations where the students spent several minutes on a problem without making any progress, tried an incorrect solution multiple times, or referred back and forth between the instructional material and the assignment without recognizing what part of the instructions they needed to use.

3.1.2 Maple

The system used by the students during the study, Maple, is a symbolic-numeric mathematical system developed by Maplesoft with contributions by researchers at various universities. Maple has several components that have been developed over various timeframes. The Maple kernel and library have been evolving continuously since the mid 1980s. The kernel supports the Maple language on which the library is built. The library supports thousands of mathematical and other related commands. There are several interfaces through which users can interact with the Maple kernel and library. The most commonly used interface is the Standard GUI, which has been in use and continuous development since about 2000. There are two interface modes in the Standard GUI. The worksheet mode, shown in Figure 6, is designed around the direct use of typed Maple commands and supports some point-and-click interaction. The document mode, shown in Figure 7, is designed to make use of point-and-click interaction and context menus, with minimal use of Maple commands. While the point-and-click interface is designed for easy use of Maple for novice users, most complex work requires extensive use of Maple commands, so learning how to use Maple commands is useful for students studying advanced mathematics. All the observations in this study were made of students using the worksheet mode, and most interaction with Maple was accomplished by using Maple commands.



```
> f := sin(x) + cos(x);
                         f := sin(x) + cos(x)          (1)
> g := diff((1), x);
                         g := cos(x) − sin(x)          (2)
> plot(g, x=-Pi..Pi);
```

Figure 6 – Maple Standard GUI – Worksheet Mode

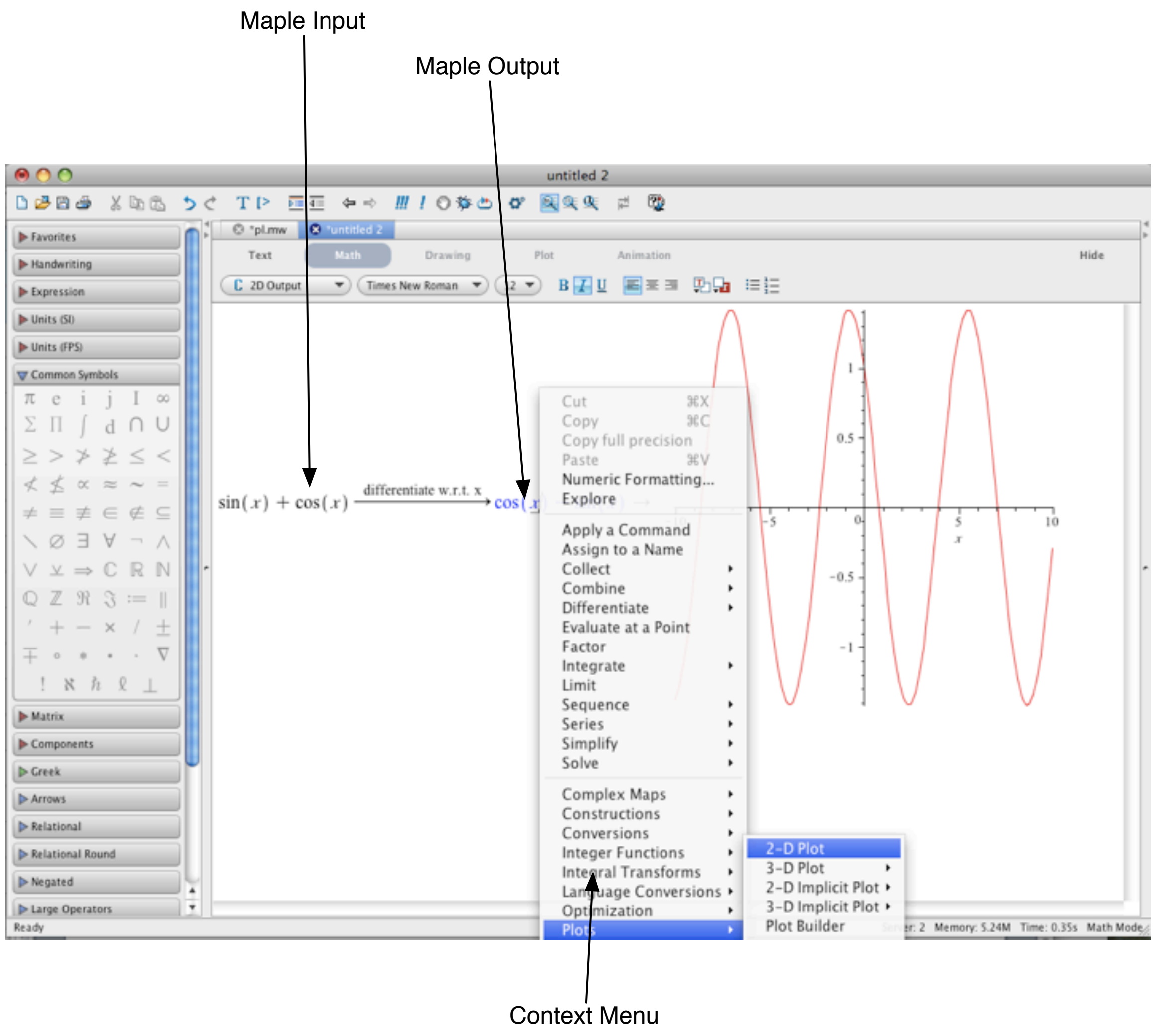


Figure 7 - Maple Standard GUI - Document Mode

3.1.3 Participants

Four students participated in this study. The participants were first-year students in the Faculty of Mathematics at the University of Waterloo. Two participants were female and two were male.

None of the students who participated in the study reported having used Maple, or any other comparable piece of software before starting their course; however, the assignment that this field study focused on was the second such assignment in the course, so the students had a little exposure to Maple when the study began. The students were supposed to complete two Maple-based assignments during the course.

3.1.4 Observed Task

Before starting any of the assignments, the students were supposed to complete an introductory Maple worksheet, and, before each assignment, they were supposed to complete a pre-lab worksheet. It is not known if, or how extensively, each student studied the introductory or pre-lab worksheets before starting the assignment; however, the students did have access to those worksheets while completing the assignment. Table 3 shows a list of Maple features covered by the material in the introductory and pre-lab worksheet for assignment 1. The students should have used those commands to complete assignment 1, which was completed before the study began.

Table 3 - Maple features that participants were expected to be familiar with from assignment 1. The relevant command names at in parentheses.

Basic arithmetic	Function definition
Variable assignment	Differentiation (diff)
Common symbols ($\pi, I, e, \sqrt{2}$)	Plotting (plot)
Numerical	Factoring
and symbolic evaluation (eval,evalf)	Equation Solving (solve and fsolve)
Simplification	

Table 4 shows the Maple features and commands introduced to the students in the pre-lab worksheet for assignment 2. The students would not have used these commands to complete an assignment before the study.

Table 4 - Maple features introduced in the pre-lab worksheet for assignment 2. The relevant command names are in parentheses.

sequences (seq)

Riemann Sums (RiesmannSum)

Newtons Method (NewtomsMethod)

Integral Approximation (ApproxInt)

3.2 Findings

During the study, as the participants completed the assignment, and notes were made, three difficulties encountered by all four students in the study were noted: understanding Maple syntax and semantics, remembering defined variables, and learning Maple and new mathematics at the same time. Each of these difficulties will be discussed in detail below.

3.2.1 Maple Syntax and Semantics

As the students tried to complete the Maple assignment, they had to learn Maple's syntax and semantics. A particular difficulty the students encountered was differentiating between a Maple function and a Maple expression. A Maple function can be entered in the form

$$f := x \to \sin(x)\,;$$

To evaluate a function for some x, say $x = 2$, a Maple user enters the command

$$f(2);$$

On the other hand, a Maple expression is entered as

$$g := \sin(x)\,;$$

To evaluate an expression for some x, again say $x = 2$, a Maple user enters the command

$$eval(g, x = 2);$$

This difference in defining and evaluating functions and expressions is confusing enough, but the difference between the two forms becomes even more subtle, and confusing, for a new user when trying to create a plot. To plot a function a user needs to enter one of two commands:

$$plot(f, -10..10)$$

or

$$plot(f(x), x = -10..10)$$

To plot an expression, the user needs to enter the command

$$plot(g, x = -10..10)$$

If, for example, a user confuses functions and expressions and enters the command

$$plot(f, x = -10..10)$$

Maple responds with the error message

```
Error, (in plot) expected a range but received x = -10 .. 10
```

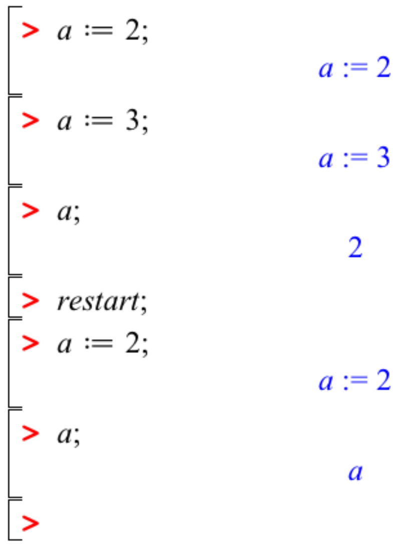
A new user who does not understand what a range is, as a Maple type, will not understand this error message or how to fix the problem.

As observed in the study, because the students did not understand the difference between a function and an expression, and why that difference might result in the errors they saw, the students were left trying to compare, character-by-character, the commands they entered with example commands in other worksheets. This character-by-character search was both time consuming and probably working-memory resource intensive for the students as they switched their focus between the command they had written and the example command.

3.2.2 Remembering Defined Variables

It was noted during the study that students had difficultly remembering or otherwise determining what variables were defined at a given time. It is sometimes difficult to determine which variables are defined because the value that a variable contains cannot always be determined from looking at the current state of the worksheet.

Maple worksheets are visually organized in a linear fashion. When a user executes a line of code the output appears on the next line. However, this visually linear organization can be misleading as the input lines can be executed out of order by simply moving the caret to any input line and pressing enter. Out-of-order execution can mean the internal state of Maple can be different from what it appears to be from looking at the worksheet. Figure 8 shows part of a Maple worksheet.



```
> a := 2;
                    a := 2
> a := 3;
                    a := 3
> a;
                      2
> restart;
> a := 2;
                    a := 2
> a;
                      a
>
```

Figure 8 - A Maple execution sequence

The results from the two lines of the form

$$a;$$

are not what a user would expect from just looking at the worksheet, but the commands have not been executed in the order that they are displayed.

Therefore, if the student does not execute the worksheet in a completely linear fashion, or the worksheet contains restarts, he or she must remember what variables are currently defined and what they represent.

Having to understand and remember which variables are defined could increase the load placed on a student's working memory, since he or she must learn how Maple manages variables behind the scenes, and track the result in his or her working memory, or learn the commands to determine their values. (Since this study was completed, a new Maple feature displays a list of current variables and their values, somewhat alleviating this problem; however, a new Maple user will still need to develop an understanding of how variables work.)

3.2.3 Learning Maple and New Mathematics at the Same Time

A third problem the students experienced was caused by the assignment having dual goals of teaching the students Maple and new mathematics at the same time. For example, one part of the lab introduced the students to the concept of numerical integration[1].

As part of this problem, the assignment instructed students to "create a sequence of 11 points $\{x_k\}, k = 1..11$ with $x_1 = -1$ and $x_{11} = 1$, corresponding to a partition of [-1,1] into 10 equal subdivisions, each of width $\frac{1}{5}$. " For this task the students were to use the seq command. The seq command takes two parameters in its most basic form: a general form for each term and a range index. For this particular problem a suitable command would be:

```
seq(-1+(k-1)/5,k=1..11);
```

In the pre-lab worksheet, the students were given one sequence example and one practice problem for sequences of a form similar to the command above.

The four students were able to figure out the seq command was the correct command to use, and they were able to figure out the second parameter, k=1..11, with little trouble, but none of them determined the first parameter, -1+(k-1)/5, without substantial help.

It was hypothesized that the students' difficulties lay in trying to accomplish two general subtasks at once:

- Understanding the notation in the question, including meaning of the notation in the question and what the final sequence needed looked like. Some students began to type x_k into the first parameter, indicating they may not have understood the meaning of the notation $\{x_k\}, k = 1..11$ found in the assignment.
- Understanding what the Maple syntax for the general form for a term should be. For example, again related to the first point, some of them began to type x_k or x[k] as the first parameter.

[1] The topic of numerical integration did not appear in the course curriculum, for their current and first calculus course at the undergraduate level (list of topics found in Appendix A). However, it is not completely certain that the students had not learned the topic in secondary school.

The above tasks may have been difficult by themselves; however, trying to complete both subtasks at once may have become too challenging. To help the students, I walked them through the tasks in the following steps:

1. Write down a sequence starting at -1, adding $\frac{1}{5}$ for each subsequent term.

   All the students could write $-1, -\frac{4}{5}, -\frac{3}{5}, -\frac{2}{5}, -\frac{1}{5}, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1$.
2. Write down the first few terms as a sum such as $-1 + \frac{1}{5}$

   All the students could write $-1 + \frac{1}{5}, -1 + \frac{2}{5}, \ldots$
3. Write down a general term for the terms in the sequence.

   With some work, all the students eventually got $-1 + \frac{k-1}{5}$.
4. Use the result from step 3 to complete the command for creating the sequence.

The students seemed to have little difficulty completing each of the above steps, and once they completed the third step, they seemed to quickly realize that the answer to that step was the Maple expression they were looking for. Following the method in Sweller and Chandler (1994), the above steps may provide an a priori estimate of interacting elements that contributed to cognitive load imposed by the problem. Therefore, the students' ability to complete each individual steps but inability to complete the whole problem without help could be an indication of high cognitive load caused by trying to integrate all the steps needed to solve the problem.

3.2.4 Limitations

There were several limitations of this study that are important to discuss before discussing the results. Because of the nature of a field study, there is inherently a lower level of validity control (McGrath, 1984). In this study in particular, the exact knowledge that students entered the study with was unknown. For instance, there was no indication in the curriculum (University of Waterloo, 2009) that the students who participated in the study had previously learned numerical integration, the mathematical topic for the last mentioned student difficulty; however, it is not certain they had not seen the topic. Furthermore, it is not known how well, or if at all, the students learned the Maple commands from the first assignment or pre-lab.

Another limitation of the study is the lack of investigation of the cognitive processes the students engaged in while completing the problems in the study. For example, in the case of trying to learn Maple and new mathematics, while the analysis is similar to the a priori assessment of cognitive load in Sweller and Chandler (1994), the cognitive process actually used by the students, and the process that could have led to a successful solution, was not determined in the study. By considering existing literature on types of knowledge used in mathematical problem solving (eg. Mayer, 2005, 2003; Kilpatrick, Swafford, & Findell, 2001), it may be possible to infer the knowledge that students used or needed to use in the problem. Alternatively, further studies, using an empirical-positivist approach (Marmaras & Nathanael, 2005), guided by the aforementioned theoretical work on mathematical knowledge, may lead to a better understanding of the cognitive processes used by the students.

3.2.5 Field Study Summary

In this section, a preliminary study was examined in which four students were observed as they completed an assignment for a first-year calculus course. During the study, three types of difficulties that the students had were noted: confusion about syntax and semantics of Maple, difficulty remembering the state of Maple variables, and difficulty learning new mathematics and Maple at the same time. Each of these problems could be explained, at least partially, in terms of working memory overload. In the case of understanding the syntax and semantics of Maple, the students had to use their working memory to compare their solution attempt character-by-character to an example from the instructional material. In the case of remembering the state of Maple and the defined variables, the students had to devote working memory resources to figuring out the state of Maple. The third difficulty, learning math and Maple at the same time, could have been caused by the students' working memory resources being overloaded, as they tried to determine and combine all the steps needed to solve the problems.

All three of the students' difficulties explored here would potentially be alleviated if the students had a better understanding of the syntax and semantics of Maple and the needed Maple commands, allowing them to use rule-based behaviour in their use of Maple. Such rule-based behaviour would reduce the cognitive demand of using Maple. There is obviously a cyclical problem: the students might perform the assignments better if they knew Maple better. However, the assignment was also designed to introduce new mathematics at the same time as Maple, and, based on the predictions of CLT (eg. Sweller et al. 2011, Clarke et al. 2005), the dual learning task may have imposed too high of a cognitive load, reducing the ability of the students to form the schemas that could drive rule-based behaviours.

In chapter 4 and chapter 5, in an attempt to solve the above problem, rule-based behaviours, and then instructional materials, for the use of Maple are developed. If students can learn to use such rule-based behaviours, the use of Maple should be possible with reduced cognitive load. The students may then be in a better position to use Maple to learn new mathematics, similar to the situation described by Clarke et al. (2005), who noted that students who had first learned a spreadsheet were better able to use it to learn new mathematics. In the study presented next, the use of Maple by experienced users was examined in order to develop a WDA and CTA for the use of Maple, which are used in Chapter 4 to develop the rules for Maple use to be taught to students.

3.3 Experienced Maple User Study

In the field study analysis, it was noted that students experienced difficulties when using Maple, possibly due to three main issues: understanding Maple syntax, particularly the difference between functions and expressions; understanding the internal state of Maple, particularly remembering the state of variables; and learning new mathematics and Maple at the same time. From these observations, it was concluded that it might be useful to teach students to use Maple, before they use the system to learn new mathematics.

In the study described in this section, to determine what to teach students, it was examined how users who are already familiar with Maple used it to solve problems and their solution methods were compared to those used by the participants when they did not have access to Maple. It is also observed what commands and syntax the participants used when solving problems using Maple. The observations are used to develop a WDA abstraction hierarchy and CTA decision ladder for Maple use when solving mathematical problems. The behaviours of the participants during the study are then analysed from the perspective of the rules that could guide the behaviour described by the CTA decision ladder, in order to determine what behaviours should be taught to students to use Maple.

3.3.1 Hypothesis

It was hypothesized that the strategies used by the participants would differ when using Maple for the problem solving from when they did not have access to Maple, because when the participants had access to Maple, they would have a number of tools available that are designed to solve certain categories of problems (such as Maple's `dsolve` and `minimize` command and Vector packages).

3.3.2 Method

This study was designed as a field experiment (McGrath, 1984), with one major factor controlled: the availability of Maple when solving the problems given to the participants. The participants each solved four questions, and for half the questions the participants were allowed to use Maple. Which questions the participants could use Maple for varied over the participants, as described below.

During the study, the actions of the participants were recorded using a video camera, when the participants answered questions on paper, or screen recording software, when the participants used Maple. The recordings of the participant's Maple or paper solutions were analysed to determine what strategies the participants used to solve the problems and what Maple commands were used.

In the following sections, the participants, the problems solved, the equipment the participants had available, and the interview questions are discussed.

3.3.2.1 Participants

For this study, employees of Maplesoft, the company that develops Maple, were recruited. An email was sent to all members of the research and development team at Maplesoft and the seven employees who responded participated in the study. The participants had worked at Maplesoft for two to eleven years and for an average of five years. Each participant had an education background in Math or Engineering and six of the participants had PhDs in Math or Engineering. All participants worked in the Research and Development department at Maplesoft and had spent time developing Maple application worksheets or doing software development on the Maple system. All the participants were male.

3.3.2.2 Problems

Each participant was asked to solve four problems. The same four problems were used for each participant with one exception (the exception is discussed below). The participants were asked to solve the problems using pencil and paper or Maple, as is discussed in the equipment section below.

The problems were designed to be challenging enough that the answers were not immediately apparent and some consideration of strategy was needed when solving them. However, due to the diversity of areas of expertise amongst the participants and the length of time since the participants would have solved common problems, the problems were kept simple. Also, the problems assigned were similar to those solved by undergraduate students learning Maple so the results would be useful for improving the teaching of Maple to undergraduate students. Finally, each of the problems had multiple possible solution types, particularly when using Maple.

The four original problems were:

1. What is the area enclosed by the curves $(y + 2)^2 = x + 3$ and $y = x - 2$?

2. What's the minimum value of $\frac{1}{4}x^4 + \frac{11}{3}x^3 + 17x^2 + 24x$?

3. What is the single force (magnitude F and angle a) that is the equivalent to F1 and F2?

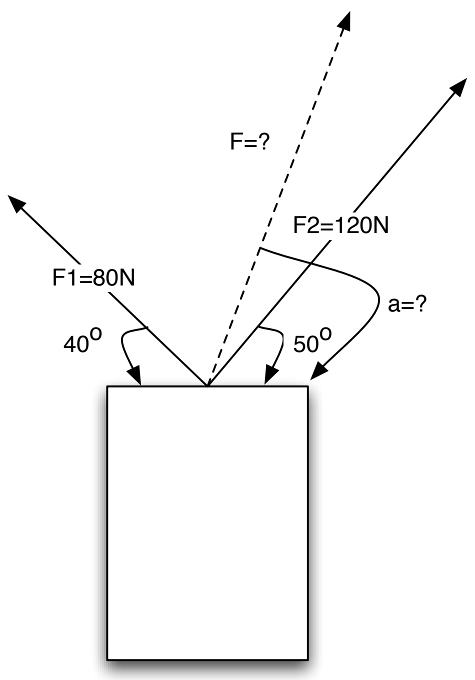


Figure 9 - Figure for problem 3 in experienced Maple user study

4. A ball is launched with an initial speed of 30m/s at an angle of 60° from horizontal. How far from the initial location would the ball land (assume no air resistance, g=9.8m/s^2)?

After the first participant completed the four problems, he suggested that the first and second problems be simplified. He thought the first problem was tricky unless the person solving it realized that the integral should be done with respect to y. Therefore, the first problem was changed to:

1. What is the area enclosed by the curves $y = (x + 2)^2 - 3$ and $y = x + 2$?
He also felt the second problem given, without hints (as he was given), would be difficult to do by hand. The modified problem was:

2. What's the minimum value of $\frac{1}{4}x^4 + \frac{11}{3}x^3 + 17x^2 + 24$?

The modified problem was easier because x was a factor of the derivative. The limitations caused by this change is discussed in the limitations section below.

3.3.2.3 Participant Equipment

The resources available to each participant to assist with the solving of the problems were:

- Paper and pencil
- Calculator (supplied or their own)
- One physics and one calculus textbook
- Maple on a windows-based laptop
- One physics text book (Serway, 1996)
- One calculus text book (Stewart, 1995)

The participants were free to use the above resources on any of the problems except for Maple. Maple was available for only half the problems. For each participant, the problems for which Maple was available for are shown in Table 5.

Table 5 - Maple use for experienced problem solving study.

Participant Problem	1	2	3	4	5	6	7
1	Maple		Maple			Maple	Maple
2		Maple		Maple	Maple		
3		Maple	Maple	Maple	Maple		
4	Maple					Maple	Maple

3.3.3 Results

This study's data was analysed by watching the videos of the participants when they solved the problems and looking at the participants' solutions, either on paper or in a Maple worksheet. A list of the steps that each participant took for each problem was made. That list is found in Appendix B.

There are a number of different strategies that could be used for each problem given. In this section, the different strategies used for each problem by each participant are examined. The various strategies used for each problem are described and then summarized in a table. Additionally, for each problem, the Maple syntax features found in the Maple manual's Worksheet Introduction section that were used by the participants who solved each problem using Maple are summarised. The syntax features used by the participants are also listed by problem in Appendix B.

3.3.3.1 Problem 1

What is the area enclosed by the curves $y = (x+2)^2 - 3$ and $y = x + 2$?

The first problem was fairly straightforward to solve, and all participants used generally the same strategy. An integral of the difference of the two functions needed to be computed between two points. There were three main steps taken to solve the problem (one step was not used by two participants).

First, six of the participants created a plot of the functions. In the case of Maple the plot command was used to draw a plot for the two curves. In the case of paper and pencil a quick sketch was drawn based on knowing the axes intercept points of the line and the bottom of the parabola. One participant, number 4, who did not have access to Maple, did not create a plot. Participant 3, who used Maple, created a plot only at the end of his solution to check his answer.

Second, the end points for the integral, the intersection points of the two curves, had to be determined. Determining the end points required solving a quadratic equation. In the case of Maple use, the participants used either `fsolve` or `solve` and `evalf` to find the intersection points. When using paper and pencil, most participants used the quadratic formula. Some participants who were not using Maple tried to factor the quadratic; however, the roots were not integers, so factoring was difficult. In order to solve the quadratic equation, one participant, participant 4, completed the Square, which was an unexpected strategy. He also used the quadratic equation to solve the equation when the values he computed by completing the Square did not satisfy the original equations. When his results from completing the Square and using the quadratic equation matched, he was satisfied that the intersection points he computed were correct.

Third, the integral of the difference between the two curves was computed. Evaluating the integral was fairly straightforward both in Maple and on paper. Participants using Maple used the `int` command. Participant 1 used an unevaluated integral to ensure the form was correct and then the `value` command to evaluate the integral. Every participant who was solving the problem on paper used a calculator to do the integral and the paper as a place to store intermediate calculations.

The participants who completed the problem using Maple used the following syntax features: function calls, lists, ranges, assignment, and equality.

3.3.3.2 Problem 2

What's the minimum value of $\frac{1}{4}x^4 + \frac{11}{3}x^3 + 17x^2 + 24x$?

Two different types of strategies were used for problem 2. One strategy, obviously only available to the participants using Maple, was to use the Maple `minimize` command. The second strategy involved finding where the derivative of the expression was zero. Because one of the roots of the derivative was zero, the participants realized that root must represent the location of the minimum. Participant 1 used a different strategy, because he had the original problem that had different roots for the derivative; he used a second derivative test.

Table 6 shows the number of participants who used the different strategies by condition. The participants in the condition where maple was not available, did not have the option of using the minimize command.

Table 6 – Number of participants who used each strategy by condition for problem two. The blank space indicated a strategy not available for a condition.

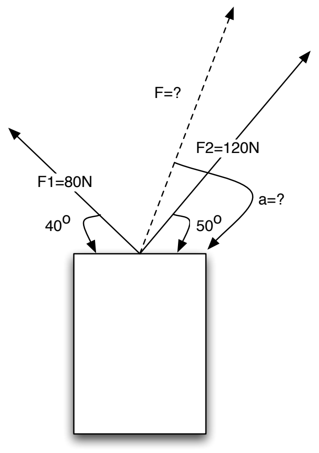
Strategy / **Condition**	Maple Minimize Command	Find Roots of Derivative
Non-Maple		4
Maple	2	1

For question two, while two Maple participants used the minimize command, a relatively simple Maple command, one participant still used the same strategy used by the participants who did not have access to Maple, such a strategy involved the use of the `diff`, `solve`, `fsolve`, and `evalf` commands.

The participants who completed the problem using Maple used the following syntax features: function calls, ranges, assignment, and equality.

3.3.3.3 Problem 3

What is the single force (magnitude F and angle a) that is the equivalent to F1 and F2?



Question three involved computing a resultant force vector as the sum of two force vectors. The strategies used for question three fit into three categories. Most of the participants used a component strategy, whereby they broke each of the vectors into their horizontal and vertical components and then added up the components. One participant used complex numbers to solve the problem. Such a strategy is conceptually similar to the component strategy, but where the real and imaginary parts of the numbers track the horizontal and vertical components of the vectors. However, a student might see these as different strategies because the Maple commands to execute the two strategies would differ. One participant used the Pythagorean theorem to solve the problem, which was a possible strategy since the two force vectors were perpendicular.

Table 7 summarizes the number of participants in each condition that used each strategy.

Table 7 - Number of participants who used each strategy by condition for problem three. The blank space indicated a strategy not available for a condition.

Strategy / **Condition**	Components	Complex Numbers	Pythagorean Theorem
Non-Maple	2	1	0
Maple	3	0	1

The use of the Pythagorean Theorem-based solution was not anticipated. The participant who used the Pythagorean Theorem was the only person who seemed to notice that the vectors were perpendicular. This realization allowed a very quick solution strategy. From Table 7, it can be seen that participants used the same types of strategies across conditions.

The participants who completed the problem using Maple used the following syntax features: function calls, lists, ranges, assignment, and equality.

3.3.3.4 Problem 4

A ball is launched with an initial speed of 30m/s at an angle of 60° from horizontal. How far from the initial location would the ball land (assume no air resistance, g=9.8m/s^2)?

The fourth problem involved computing the distance a projectile travels when launched at a given angle and speed. There were two basic strategies used. Each participant either solved a differential equation or used a known dynamics equation. All participants, except participant number one, drew a sketch on paper when starting the problem. A sketch was drawn on paper even by those who used maple to solve the problem.

The differential equation strategy involved using a differential equation to compute the time needed for the object to return back to the ground. Two participants used a second-order differential equation starting with $a(t) = a_0$ and one used a first-order differential equation starting with $v(t) = v_0 + at$. a_0 represents the acceleration due to gravity, v_0 is the initial velocity as given in the problem. The participants using Maple used the Maple int and eval commands to solve the differential equations.

Interestingly, no one used `dsolve`, the Maple command to solve differential equations. Some participants commented that they did not know how to use `dsolve`. One participant commented that he could have used `dsolve` to check his answer.

Some of the participants used known formulas to solve the problem. Participant 1 used a formula he looked up in the provided physics textbook (Serway, 1996) that gave the range of a projectile given the initial angle and speed to be

$$R = \frac{v_0^2 \sin(2\theta)}{g}$$

Other participants knew the formula for distance travelled

$$d = v_0 t + \frac{1}{2} a t^2$$

and used it, along with the vertical and horizontal components of the initial velocity, to compute at what time t^* the object would hit the ground again and then computed the range using $R = v_{0\,horizontal} t^*$.

Table 8 summarizes the strategies used for each condition.

Table 8 - Number of participants who used each strategy by condition for problem four.

The blank space indicated a strategy not available for a condition.

Strategy / **Tool**	Known Equation(s)	Differential Equation	dsovle
Non-Maple	2	2	
Maple	2	1	0

The participants who completed the problem using Maple used the following syntax features: function calls, assignment, and equality.

3.3.3.5 Commands and Syntax Used

In this section the commands and syntax used by the participants are listed. Table 9 lists the commands used by the participants.

Table 9 – Commands used in experienced Maple user study

Command	Use
help	Get help on Maple commands
plot	Create a plot of a function or expression
diff	Differentiate an expression
int/Int	Integrate an expression
value	Evaluate an unevaluated integral
solve	Symbolically solve an equation
fsolve	Numerically solve an equation
minimize	Find the minimum of an expression or function
select	Select the operands of an expression that match a condition
remove	Select the operands of an expression that do not match a condition
trigonometry functions (sin, cos, tan, atan)	Compute various trigonometry functions
sqrt	Compute Square root
Units	Allow use of degrees in functions such as sin
eval	Evaluates a function at a point

Most of these commands are basic Maple commands, appearing in the list of top commands in the Worksheet Introduction section of the Maple manual. Furthermore, most of these commands have a direct correspondence with a method used in solving problems on paper. There were a few more-advanced Maple commands used such as `select`, `remove, and minimize`; however, all the problems could be solved without these commands and most participants did not use them.

The participants also used a number of Maple syntax features found in the Worksheet Introduction section of the Maple manual to complete their solutions in Maple. Those features are summarized in Table 10.

Table 10 – Maple syntax features used in experienced Maple user study

Feature	Example	Use
Function Calls	plot(x)	Performing a function on some objects
Assignments	g := x+2	Assignment to a variable
Equality	x = 2	Specifying a equation to solve or a point to evaluate at
Ranges	3 .. 4	Specifying a range to perform an operation over
Lists	[a,b]	Grouping objects

3.3.4 Discussion on Strategies and Commands

The results of this study, along with a consideration of the purposes of Maple, were used to develop a WDA and CTA for the use of Maple. In this section, a WDA and CTA are presented, and then the results are discussed in terms of that analysis.

3.3.4.1 Maple Work Domain and Control Tasks Analyses

Figure 10 shows a WDA abstraction hierarchy for solving math problems using Maple. In the abstraction hierarchy, the functional purpose is simply to solve math problems. The abstract functions are the principles that must guide the solving of those problems, which is the correct following of mathematical principles. The principles can be achieved by providing the correct input to Maple, using various algorithmic solutions, and interpreting the output from Maple, as shown in the general functions. Each of the general functions is achieved by a combination of Maple commands that define certain Maple calculations, as shown in the physical functions. Finally, the physical form level shows that Maple syntax is used to specify the commands to Maple, and that the syntax that specifies a Maple command can be decomposed into parts including the command name, parameters, numeric values, and variables. Example command names from Table 9 and example parameter syntax from Table 10 are shown in the abstraction hierarchy.

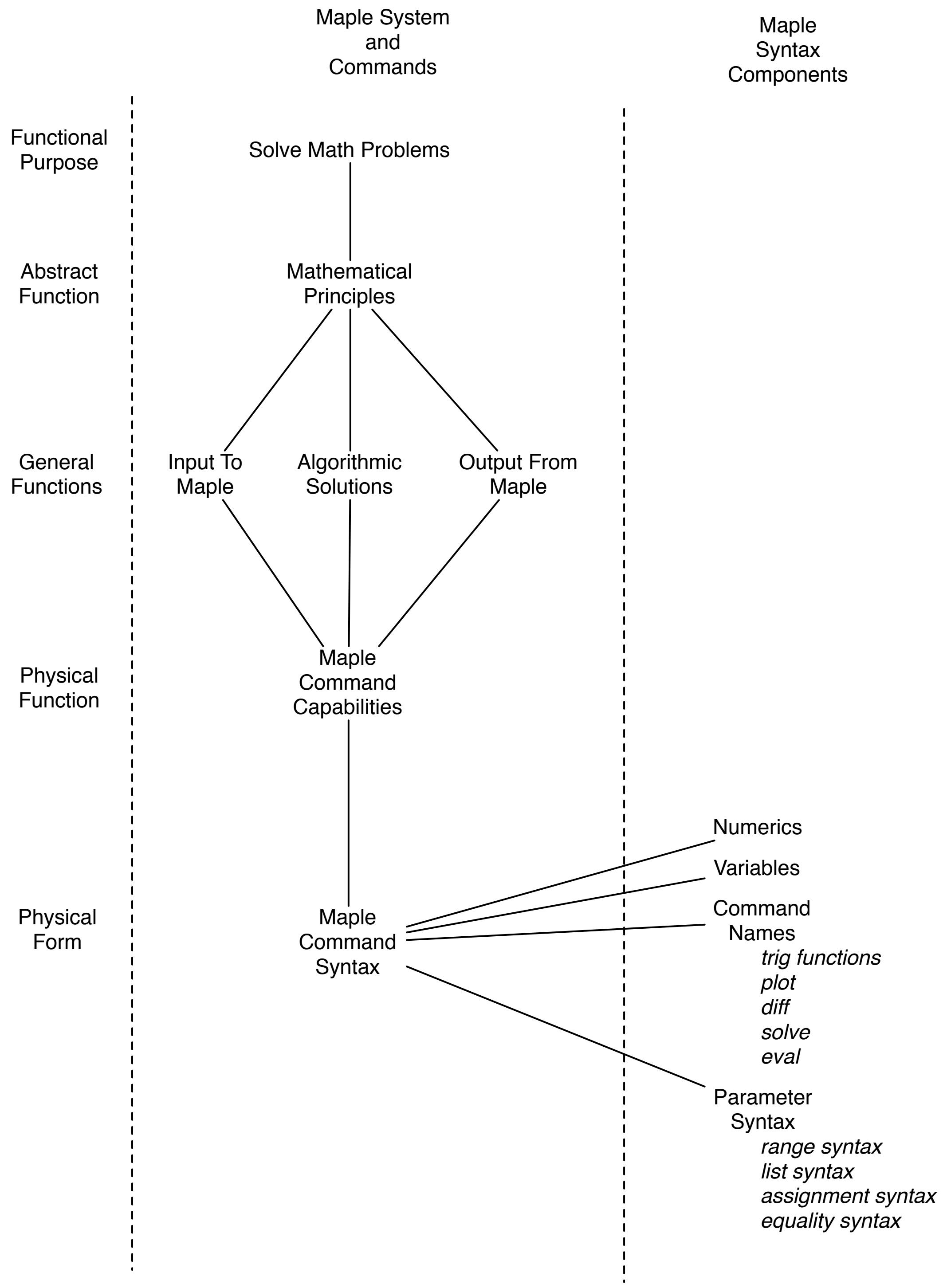
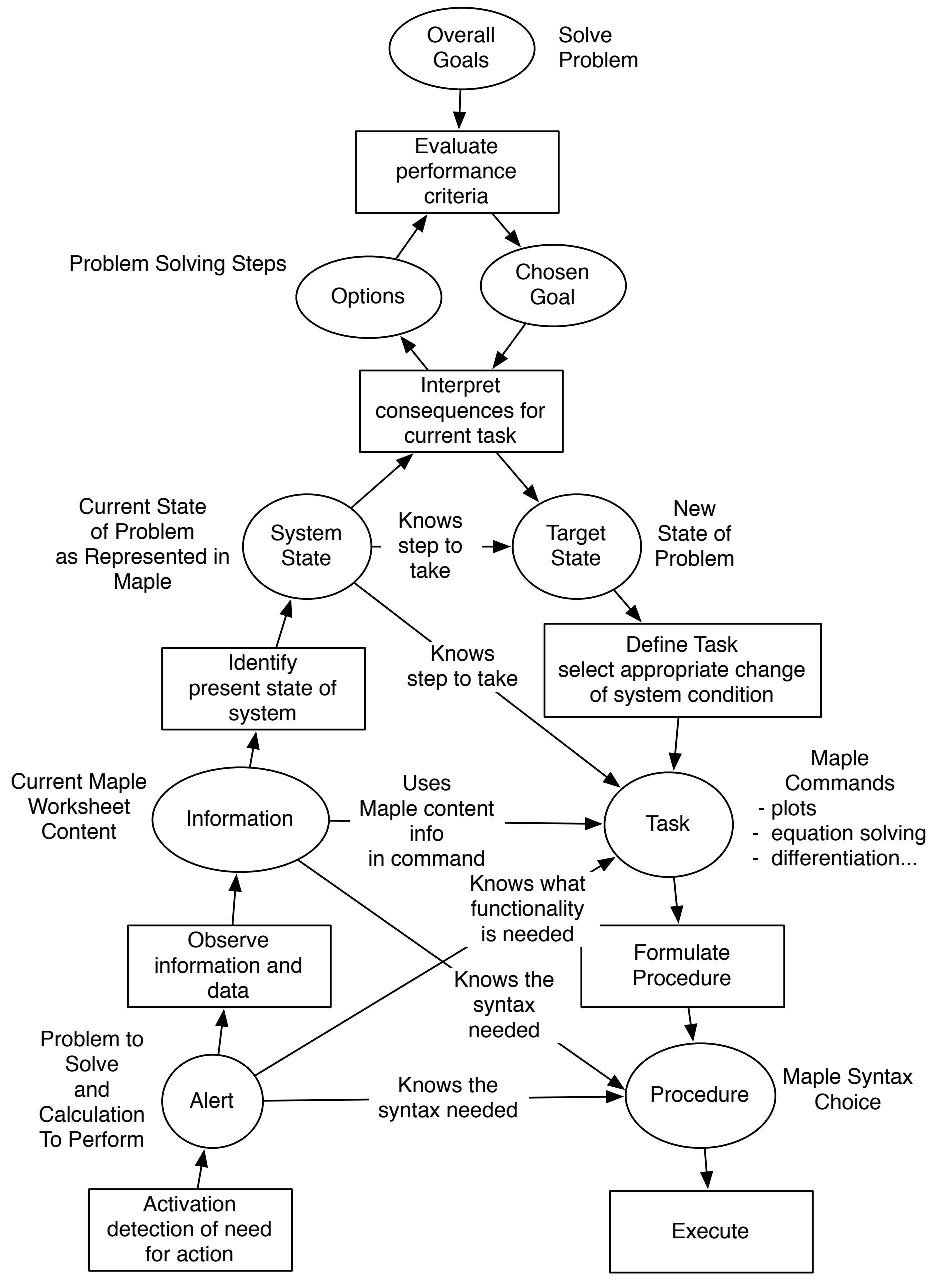


Figure 10 - WDA for using Maple to solve (parts of) math problems

To consider how an operator might actually use Maple and make decisions about what Maple syntax to enter into Maple to use certain commands to solve problems in the work domain described in Figure 10, the decision ladder in Figure 11 can be considered. The decision ladder shows steps that might be involved in an operator's use of Maple, during which he or she modifies the state of the Maple work domain, and the following description shows the corresponding states in the decision ladder in parentheses. After starting with the detection of a need to perform a calculation to solve part of a math problem (alert), the operator might immediately know what syntax is needed and jump right to a solution procedure (procedure) that involves the entry of known Maple syntax. Alternatively, the operator might know what Maple functionality is needed (task), represented by Maple commands in the WDA, allowing him or her to jump to the task and from there formulate the procedure. If, however, an operator needs to perform a calculation that involves previous Maple calculations, the operator might then need to gather information from the worksheet (information). Using that information, the operator might then be able to determine the Maple commands (task) or syntax (procedure) needed that could involve considerations of the current state of Maple.

If the operator does not know what action to take based on the content of the Maple worksheet, he or she might need to consider the state of the problem (system state). From knowing the state of the problem, the operator might know a new problem state to move to or may need to evaluate problem-solving steps (options) that could be taken. Following a decision on a problem-solving step, the operator will need to decide on a Maple command (task) and Maple syntax (procedure) to implement that command. The above describes the structure of the decision ladder and some possible paths that could be taken through the ladder. In the next section, the strategies of the participants during the study is described and then considered in the context of the decision ladder.



Overall Goals
Solve Problem
Evaluate performance criteria
Problem Solving Steps
Options
Chosen Goal
Interpret consequences for current task
Current State of Problem as Represented in Maple
System State
Knows step to take
Target State
New State of Problem
Identify present state of system
Knows step to take
Define Task select appropriate change of system condition
Current Maple Worksheet Content
Information
Uses Maple content info in command
Task
Maple Commands
- plots
- equation solving
- differentiation...
Knows what functionality is needed
Observe information and data
Knows the syntax needed
Formulate Procedure
Problem to Solve and Calculation To Perform
Alert
Knows the syntax needed
Procedure
Maple Syntax Choice
Activation detection of need for action
Execute

Figure 11 – Decision Ladder for Entering Maple Syntax

3.3.4.2 Participant Strategies

Appendix B provides details of the strategies used by each participant for each problem. From the summary of the types of strategies for each problem, it can be seen that in many cases participants used the same types of strategies regardless of if they completed a problem with access to Maple or not. It was hypothesised that the participants would make more use of Maple commands such as `dsolve` and `minimize` where appropriate, as these commands could be used to solve the problems in one step, but participants used only the most basic Maple commands, with only a few exceptions. Even though the more complicated commands could provide shorter or faster solutions, people generally used strategies that were simple.

After completing the study, participant 3 gave some comments that provided some insight as to why he, and possibly the others, avoided more advanced commands. He stated that he would rather work from first principles, even when using Maple, for a number of reasons. First, it is easier to remember the usage for the simple commands than all the options for complicated commands, and it is easier to remember the general methods of how to solve a problem. Second, it is easier to trust the simple commands and trust that he understands what they are doing. Third, it is often faster to use more of the simple commands than look up the more complicated commands.

The participants used Maple commands to complete some of the more tedious parts of the familiar solution strategies. In particular, basic Maple commands, such as `solve` and `fsolve`, were used to solve quadratic equations, a task the participants found rather tedious to complete without Maple. The `int` command was also particularly useful, again allowing participants to avoid the tedious task of solving definite integrals. Finally, the `plot` command was helpful. While the participants did not need to create plots as part of the solutions for any of the problems, they found plots helpful in understanding the problems and solutions. Some participants created plots sketches on paper; however, creating plots in Maple was much faster.

In addition to using basic Maple commands, many of the participants implemented solution strategies that would still be familiar to people familiar with the math problems but who had not used Maple, for example participants used similar strategies for problems 1, 3, and 4. However, there was one exception to using the familiar non-Maple strategies in Maple: the use of the `minimize` command for problem three by two participants. Participant 2 did have to do some experimentation to remember how the minimize command worked and what the meaning of its output was, demonstrating the problem that participant 3 noted with remembering how the more advanced commands worked.

The behaviour of using familiar strategies and Maple can be understood in terms of the decision ladder in Figure 11. The use of familiar commands could allow the leaps from the alert, information, or system state information states to the Maple task or procedure when participants could recognize familiar Maple worksheet states. An operator would be able to make such a leap if they knew an appropriate rule-based behaviour for using a familiar command, and if an operator did know such a familiar behaviour, he or she may be prone to use it (Reason, 1990). However, if a participant needed to consider problem solving step options, they may consider options for solving a problem that he or she is already familiar with, again due to humans' tendency to use familiar behaviours (Reason, 1990).

3.3.5 Limitations

There are several limitations of this study. First, the limited number of participants did not allow a statistical analysis of the results.

Second, participant 1 solved different version of problems one and two. The participant could be excluded from the study as a pilot; however, as this study is being used to explore the use of Maple to solve problems, his data concerning using of Maple commands and syntax is still useful. Removal of participant 1 would not change the general conclusions of this study.

Third, the design of this study did not take into account effects of the ordering of the questions. All participants completed the problems in the same order, and it is possible that the solution a participant used for one problem could have influenced the solutions he or she used for subsequent problems. Therefore, a future study of this nature could benefit from a randomization of problem-presentation order to help detect order effects.

Finally, the problems used in this study were fairly simple to solve and could be solved with relatively simple commands, which is generally how the participants solved the problems. A future study involving more difficult problems could reveal when and how Maple users would use more complex commands.

3.3.6 Summary

This study looked at the difference in behaviours when participants use Maple to solve math problems and when they used paper and pencil. Additionally, the Maple commands and syntax used were noted.

From an analysis of the use of the Maple system, a WDA abstraction hierarchy and CTA decision ladder were developed. The use of solution strategies that would be familiar on paper and the use of basic Maple commands were analysed in terms of the decision ladder and explained by humans' tendency to use familiar behaviours. However, regardless of how the operator ultimately reaches the knowledge state in the decision ladder of Maple syntax, an operator will need to either formulate or know the Maple command capabilities and syntax for a task. Therefore, there is an opportunity for an operator to learn rules that will allow him or her to generate Maple syntax for Maple tasks and make it more likely that an operator will successfully use Maple, and employ fewer cognitive resources in doing so.

In the next chapter, possible rules that would allow operators to know Maple command capabilities and syntax are developed from the WDA and CTA presented in this chapter. Then in Chapter 5, those rules are used to develop training material, with the goal of training students to become familiar with the Maple commands and be able to use them to solve familiar parts of math problems using Maple.

Chapter 4
Identifying and Training Transferable Rules

In novel and unexpected situations, when operating a system, an operator will have to engage in model-based behaviour, which can be cognitively demanding. With the recognition that most tool use in a system involves a mixture of behaviour types, a reduction in the cognitive demand of a task could occur if an operator could use rule- and skill- based behaviour as much as possible, because those types of behaviours generally impose a lower cognitive demand (Vicente, 2002; Vicente, 1999). Particularly when completing novel tasks, where extensive model-based behaviour is typical, an operator who can use rule and skill based behaviour will complete the task more cognitively efficiently. Therefore, it could be useful to train operators to use rule-based behaviours that can be reused as part of the model-based behaviour engaged for completing novel tasks. The first step in accomplishing this objective is to develop methods for identifying rule-based behaviours.

4.1 Determination of Rules

Before training of rules can occur, the rule-based behaviours for tool use that can be transferred to novel scenarios must be identified. I propose the use of the SRM Taxonomy and inventory method to identify what rule-based behaviours an operator could learn in order to use a tool efficiently. In completing a SRM inventory, an analyst determines for various information-processing tasks methods of completing those tasks using various types of behaviours (Kilgore, St-Cyr, & Jamieson, 2009). In this section, I examine what rules could be derived that might be helpful for learning.

Because a goal of CWA is the development of descriptions of systems that allow flexible, but efficient, behaviours in unexpected situations, the rules that are developed should be ones that help operators act efficiently in new situations. Therefore, the rules should be ones that are transferable to new situations. By combining ideas from existing work on rule-based behaviours (e.g. Vicente, 1999; Rasmussen, 1983, 1986; Reason, 1990), with the literature on training transfer (e.g. Bransford & Schwartz, 1999; Day & Goldstone, 2012), I developed three guidelines to guide the development of rules that might be useful in new situations. The proposed guidelines are related to sign recognition and action mapping, discernment of appropriate context, and preparation for chaining and generalizing.

4.1.1 Guideline One – Sign Recognition and Action Mapping

Rule-based behaviours are useful because they tend to impose less cognitive load than model-based behaviours (Rasmussen, 1989; Vicente, 1999). Furthermore, they tend to be executed with fewer errors relative to the opportunity for error (Reason, 1990). In new and unexpected situations, an operator will need to act using model-based behaviour; however, as part of that model-based behaviour they can engage some rule-based behaviour (Rasmussen, 1986). In order to use such rule-based behaviours the operators will need to recognize the appropriate signs indicating familiar parts of situations where known rules can be used (Rasmussen, 1983). Day and Goldstone (2012) claimed that the recognition of familiar perceptual features, which could include signs, is an important trigger of transfer. Therefore, it is possible that an operator could recognize a known sign as part of a task and apply a previously learned rule.

Vicente (1999) and Divalou and Marmaras (2009) recommended the support of rules through interface designs that provide a clear mapping to the work domain and help operators recall previous experiences. Similarly, the first guideline for rules for training can be stated as follows:

1. *An operator should have schemas that allow him or her to recognize signs during the operation of a system that indicate familiar tasks, or sub-tasks, and map those signs to appropriate rule-based behaviours.*

What parts of the system the rules should cover the operation of can be defined by examining a WDA abstraction hierarchy and CTA decision ladder as Naikar (2009, 2006b) has already recommended as a method for considering training. Similar to her ideas, the rules should be related to the various objects and concepts in the work domain as represented in an abstraction hierarchy and the relationships between them, as well as the tasks and shortcuts in the decision ladder. Examples of rules developed by examining the abstraction hierarchy and decision ladder are given later in this chapter.

4.1.2 Guideline Two – Discernment of Appropriate Context

In addition to using the objects and concepts in the work domain, an operator must also understand the constraints of the system that must be respected (Vicente, 1999). As Naikar (2009) pointed out, during training, operators must learn to understand the system constraints and how to deal with constraint violations. Moreover, operators need to know when the rules that they might want to use are inappropriate in a certain context, so that the application of rules does not cause errors (Olsen &

Rasmussen, 1989; Reason, 1990). Therefore, if operators are going to learn the rules related to the use of objects, concepts, and relations in the work domain, they must also learn to use only those rules that are appropriate in a given context in order to avoid violating system constraints. Bransford and Schwartz (1999) and Schwartz et al. (2005) claimed that an important part of transfer is to be able to distinguish one situation from another so that the proper knowledge can be applied. From this discussion, the second guideline of training rules can be summarized as follows:

2. *An operator needs to be able to distinguish valid contexts and tasks for his or her known rule-based behaviours from invalid contexts or tasks and know how the applications of rules will affect the system with respect to its constraints. As a consequence, the operator needs to be able to distinguish between two rules that may be similar but have different levels of appropriateness to a given situation.*

To determine what appropriate context for a rule is, the WDA can be considered. As Naikar (2009, 2006b) pointed out, the work domain abstraction hierarchy can be used to consider what constraints exist in a system that should not be violated, as could happen if a rule was used in an inappropriate context.

4.1.3 Guideline Three – Preparation for Chaining and Generalizing

Rules could be useful in promoting further rule-based behaviours. Based on the discussion on transfer, the rule-based behaviours identified should be ones that set an operator on a learning trajectory such that he or she can continue to learn through transferring the rules to novel situations (Bransford & Schwartz 1999).

There are two manners in which rules could serve the purpose of placing an operator on a learning trajectory. First, rule-based behaviours can be undertaken by chaining together various rules (Rasmussen, 1983). Therefore, rules that may serve as components of future rules may also be of use. Second, an important part of future learning is the recognition and understanding of generalizations (Day & Goldstone, 2012). Generalizations of rules could be related to more abstract concepts in the WDA abstraction hierarchy, with more abstract concepts being at higher levels in the hierarchy (Rasmussen, 1985).

The third guideline for training rules can be summarized as follows:

3. The operator needs rules that will promote future learning including rules that will be useful as part of other future rules and examples of more general rules.

The abstraction hierarchy from a WDA may be useful in developing rules that fit guideline three. In particular, the abstraction hierarchy means-ends and part-whole relations can help with consideration of examples of generalizations and rules that are related to sub-parts of a larger rule. Furthermore, potential leaps and shunts in a decision ladder may demonstrate some potential rules that will be useful as part of a larger task.

The above three guidelines represent very general considerations that may make rules useful for transfer in the sense of preparation for future learning. As with any analysis done with the CWA framework, the usefulness of the guidelines and the exact method for deriving the rules will depend on the domain details. Some example rules in a domain are given below.

4.1.4 Example Rules

To look at examples of rules that fit the above guidelines, some examples from the software Maple are considered. Any tool, such as Maple, has a set of interaction techniques that a person must know in order to use the tool correctly. When Maple is used to solve problems, a set of knowledge about the commands and syntax (such as how command parameters are given) is needed. In Chapter 3, the behaviour of the participants in the experienced Maple user study was discussed in terms of the CTA decision ladder. It was noted that the participants' tendency to use familiar problem-solving strategies could be explained in terms of humans' tendency to use familiar rules. Furthermore, it was noted that it might be beneficial to teach students the rules related to Maple commands and syntax so that they can form the ability to make the shortcuts described in the decision ladder and, as a result, use fewer cognitive resources in the operation of Maple. To that end, in this section, the WDA abstraction hierarchy (Figure 10) and CTA decision ladder (Figure 11) from Chapter 3 are used to develop a number of rules for the use of Maple commands and syntax that match the three guidelines for transferable rules. Such rules will, in theory, place students on a learning trajectory towards learning Maple commands and syntax.

A few examples of the derivation of possible rules and the usefulness of the rules in the context of the three guidelines will be discussed, and then a list of rules is given in Table 11. Because of the multitude of ways in which various behaviours can be combined, exactly what rule-based behaviour or combinations of behaviours an operator could use for each example will vary with his or her knowledge and interpretation of a situation. Therefore, each rule given could actually combine a number of rules.

1. As a first example, consider the task of generating a plot over a certain range and the rules needed to specify the syntax for the range of the plot. As part of a plot command, the operator must use the appropriate parameter syntax for a range, for example x = -2 .. 2. Knowledge about a rule for the range for a plot could help a student recognize the ability of Maple to plot over a range and use that feature when such a plot is required by a problem, and he or she could be more likely to know how to translate that range into Maple syntax. Furthermore, in some commands, a single point (of the form x = 2) is needed rather than a range, so the schemas for this rule may help differentiate between the two situations. Additionally, other Maple commands use the same syntax for ranges, so this rule could help with learning rules related to other commands when ranges are needed.

2. As another example, consider the rules needed for the syntax for evaluating an expression at a point. For example, an operator might want to evaluate the function sin(x) at 17. To accomplish that task, he or she would need to use the command syntax eval(sin(x), x = 17). In using this syntax, the operator would need to recognize the command name needed and the parameters needed. Additionally, when evaluating a function at a point, the second parameter is a point rather than a range, as in the previous example, so knowing this rule may help in differentiating the two types of syntax and avoiding the misapplication of either type of syntax. Finally, the syntax x = 17 is used in other commands, so the relevant rule may be reused or help with future generalizations.

3. As a final example, consider the rules needed to generate the syntax to solve an equation. An operator in this situation will need to use a number of Maple concepts to enter the appropriate command, for example solve(x^3 = 17, x). The operator will need to use the correct command name and the correct parameters. A student knowing rules for solving equations should be able to recognize that Maple can solve an equation when a problem requires it. The schemas for the rules could also help an operator distinguish the need and process for solving an equation from the process in the previous example of evaluating a function at a point.

Many more rules could be developed from an analysis of an abstraction hierarchy and decision ladder. Table 11 shows 12 example rules (including the three rules from above). For each rule, the table shows a description of the rule for an information-processing task, an example of Maple syntax that would be entered into Maple as a result of executing the rule, how the rule helps with recognition of signs and knowledge of actions, how the rule can help with knowing if the rule is appropriate to a task, and how the rule might be helpful in possible future learning. Not all rules serve all purposes immediately, but further learning could make the rule serve other purposes, hence the uses of the rules given are only examples. All examples of rules given could be used as part of future problems, reducing cognitive demand during the model-based behaviours used for those problems. Thirteen examples of rules are given, but many more examples could be developed.

Table 11 - Example rule-based behaviours for using Maple and reasons why the rules are useful based on the three guidelines discussed in this paper.

Rule Description	Relation to Abstraction Hierarchy and *Decision Ladder (in italics)*	Example Syntax	Recognize Signs	Appropriate Context	Chaining and Generalizing
Form of a range in a plot command	Command Capability Command Syntax Parameter Syntax *Task* *Procedure*	plot(x, **x = -2 .. 2**) The bold x = -2 .. 2 indicated the syntax for a range in Maple.	Recognize the ability of Maple to plot over a certain range and how to translate that range into Maple syntax.	In other rules, a single point is needed (see rules 6 for example), so the operator needs to differentiate between points and ranges.	Ranges are used in other commands and tasks, such as numerical solving, so the knowledge of this rule could be used as part other commands.
The return value of the diff command	Command Capability Command Syntax *Task* *Procedure*	diff(f,x)	Recognize that if an expression for the derivative of an expression is needed, the diff command will supply give the desired result.	As operators learn other commands, such as diff(f,x=1), which will return the derivative at x = 1, they will have to differentiate between commands that give the general form of the derivative and the derivative at a point.	Other Maple commands, such at int (integral), return expressions, so this rule will help with the understanding of other commands.

Rule Description	Relation to Abstraction Hierarchy and *Decision Ladder (in italics)*	Example Syntax	Recognize Signs	Appropriate Context	Chaining and Generalizing
Multiple expressions to a command	Command Capability Command Syntax Parameter Syntax *Task* *Procedure*	plot(**[a,b]**, x=1..3) the bold **[a,b]** shows the syntax for multiple expessions	Recognize that the plot command can plot multiple functions and how to do so.		Other commands, such as solve, can take multiple expressions, so this rule can be reused for other commands.
The syntax needed to assign an expression to a variable	Command Capability Command Syntax Parameter Syntax Variables Options *Task* *Procedure*	f := sin(x)	Recognize how to assign a value, in this case an expression, to a variable.	Differentiate assignment (:=) from equality in an expression (x = 17)	Other values, such as numbers, can be assigned to variables, so this rule may serve as an example of a more general rule for assignment.

Rule Description	Relation to Abstraction Hierarchy and *Decision Ladder (in italics)*	Example Syntax	Recognize Signs	Appropriate Context	Chaining and Generalizing
The syntax needed to evaluate a function at a point	Command Capability Command Syntax Parameter Syntax *Task* *Procedure*	eval(sin(x),**x = 17**) The bold x = 17 indicates a point in Maple.	Recognize the need to evaluate an expression at a point and the command and syntax needed. Also recognize how to indicate a point in Maple.	As per rule 2 example, in this situation, the x=17 syntax for a point rather than a range is needed. Additionally, see next rule for a differentiation with finding where an expression has a certain value.	The syntax x = 17 is useful in other Maple command and so this rule may be reused.
The syntax needed to find where an expression has a certain value.	Command Capability Command Syntax Options *Task* *Procedure*	solve(x^3=17,x)	Recognize how to find where an expression has a value.	Differentiate the processes and command for finding where an expression has a value from evaluating a function at a point (i.e. rule 6).	

Rule Description	Relation to Abstraction Hierarchy and *Decision Ladder (in italics)*	Example Syntax	Recognize Signs	Appropriate Context	Chaining and Generalizing
The syntax needed to plot an expression	Command Capability Command Syntax *Task* *Procedure*	plot(sin(x), x=1..10)	Recognize the syntax for plotting an expression	As the operation learns about functions and their difference from expressions, they will need to learn and differentiate different forms of the plot command.	Other commands, such as sum, will have similar forms as plot in so rules for the types of parameters may be abstracted.
Solve equations	Command Capability Command Syntax *Task* *Procedure*	solve(x^2+1=4,x)	Recognize the need to solve equations and how to do so.	Recognize that this process is the same as in rule 7.	Solving an equation can be used part of a larger problem, as in rule 10.

Combine processes, such as combine finding a derivative with solving an equation.	Command Capability
	
	Task
	Procedure
		f := diff(sin(x)+cos(x),x);
		solve(f=0,x)
			Recognize the need to combine multiple commands and use intermediate results.		Complex problems will often require the use of intermediate results assigned to variables

Find the derivative of an expression.	Command Capability Command Syntax *Task* *Procedure*	diff(tan(x),x)	Recognize how to take a derivative of a function.		This rule can be used as part of rules such as the previous rule.

Rule Description	Relation to Abstraction Hierarchy and *Decision Ladder (in italics)*	Example Syntax	Recognize Signs	Appropriate Context	Chaining and Generalizing
Create a plot of an expression.	Command Capability Command Syntax *Task* *Procedure*	plot(sin(x),x=-1..1)	Recognize the need to plot a function and the syntax for doing so.		
Create a plot of two expressions.	Parameter Syntax *Task* *Procedure*	plot([sin(x),cos(x)],x = -2..2)	Recognize the needed to plot two functions and the syntax for doing so.		Some of the syntax will be reusable when accomplishing other tasks such as solving simultaneous equations.

The above table shows a number of rules, and a large system would have many such rules. Once rules are derived using a SRM inventory, methods will need to be developed to help the operators learn the rules.

4.2 Teaching Rule-Based Behaviours

In the previous section, guidelines for the development of rules that may be useful for promoting future learning in new and unexpected scenarios were considered. The use of the abstraction hierarchy from a WDA and decision ladder from a CTA was suggested as a method for developing specific rules corresponding to those guidelines. Once rules for teaching are identified, methods to teach those rules are needed.

The first guideline was that the rules should allow recognitions of signs in the environment, indicating that certain rules can be applied, and knowledge of the appropriate actions given those signs. Previously the connection between schemas and rule-based behaviours was made where by schemas allow recognition and action for particular situations (Sweller, 1988), driving rule-based behaviours. Methods such as worked examples and completion problems can be useful for forming the schemas needed to recognize situations (e.g. Sweller et al., 2011).

The second guideline was the ability to detect appropriate contexts for the application of rules and knowing how the application of rules will affect system constraints. As Bransford and Schwartz (1999) noted, recognizing the applicability of knowledge is an important aspect of the preparation for future learning perspective. Day and Goldstone (2012) pointed out the need to study examples to learn the appropriate contexts for knowledge applicability, and Bransford and Schwartz (1999) further stated that the study of contrasting example in particular is important in the development of knowledge that allows the applicability of knowledge to be different as new learning occurs.

The third guideline was the generalization and chaining of rules. Reed (1993) claimed that examples are useful to help operators see patterns and sub goals, and such patterns and sub goals may lend themselves to chaining of rules. As for forming generalizations, again, Day and Goldstone (2012) recognized the need for examples, particularly dissimilar examples.

The above shows the usefulness of examples as a method of teaching rules. As noted previously, training for transfer is best done with examples that represent a whole-task context (eg. Day & Goldstone 2012; van Merriënboer 1997). However, when using contextual material, there is a risk of cognitive overload, resulting in little learning (Day & Goldstone 2012; Sweller et al. 2011).

Therefore, cognitive load must be controlled during training, using methods such as those discussed in section 2.7.4, such as completion problems and worked examples, or through the method recommended by Wickens et al. (2012) of using variable priority training (Gopher, et al., 1989), or using combination of these methods.

4.3 Summary

In this chapter, the issue of how training can be directed towards the development of rule-based behaviours that can be used as part of model-based behaviours in unfamiliar situations, allowing more efficient behaviour, was explored. By combining the existing literature on the training of rules and training transfer literature, three guidelines for potentially transferable rules were developed. The three guidelines are related to the recognition of familiar signs and mapping those signs to actions, detecting appropriate contexts for the rules, and using those rules as part of future rules or as examples for more general rules. The WDA and CTA for Maple use, developed in Chapter 3, were used to develop example rules for the use of Maple, showing that existing CWA phases can be used to determine rules for training.

It was then argued that whole-task examples might be useful in training such rules; however, it was noted that whole-task examples require a context that can potentially impose a high cognitive load, resulting in little learning. In particular, CWA is most often used in complex environments, making the cognitive load imposed by context an important concern. Therefore, in any training approach based on CWA, cognitive load must be controlled. An example of a training process, with cognitive load controlled, is presented in the next chapter. In particular, the rules developed in Table 12 of this chapter are used to develop training materials where cognitive load was controlled and evaluated the perceived workload involved in using these materials.

Chapter 5
Rule-Based Instructional Study

This third study was designed to test the hypothesis that people learn rules better when the instructional material imposes less cognitive load. Again, Maple was used for this study, allowing the rules from the CWA of Maple and findings from the first two studies to be incorporated.

As discussed, the use of context and examples in the design of instructional material can help with the motivation for learning and the understanding of rules. However, as seen in two above-mentioned studies (Clarke et al. 2005 & Robinson and Burns 2009), context, such as the unfamiliar mathematics used during instruction for those studies, sometimes interferes with learning, by increasing cognitive load. Therefore, the challenge is to ensure that the contextual material, for training the rules identified above, involves whole tasks in order to provide appropriate context (van Merriënboer, 1997), but imposes as little cognitive load as possible (van Merriënboer, Clark, and Croock 2002). Therefore, I tested a pre-training method to reduce cognitive load that is based on the recommendations of Clarke et al. (2005), Sweller et al. (2011), van Merriënboer (1997), and Gopher et al. (1989). In this method, the participants are pre-trained to use Maple while solving problems involving familiar mathematics. Such a method, while still using whole-task learning, allows the participants to focus their cognitive resources, and hence schema formations, on tasks germane to the use of Maple for the information-processing tasks listed in Table 11. This focus potentially allows the participants to learn the Maple-related rules better than if they needed to learn new mathematics at the same time. The work of Clarke et al. (2005) showed that the knowledge learned during such pre-training can be useful when solving future math problems.

5.1.1 Hypothesis

In this study, two sets of mathematical problems were used as context when teaching Maple concepts to the participants. One set involved mathematical concepts that the participants were not familiar with and the other set, intended to imposed a lower cognitive load, involved concepts that the participants were familiar with. These two sets of instructional materials defined the two conditions for the study: unfamiliar and familiar math. It was hypothesized that, because the participants in the familiar condition would not have to learn new mathematics, those participants would not have to focus as many cognitive resources on learning mathematics, allowing them to focus on the task of learning rules related to the use of Maple and, therefore, learn more of those rules. As a result of better learning of the rules, it was expected that the participants in the familiar condition would score better on a test of the rules and experience less workload while completing the test. Such a hypothesis would be consistent with the work of Clarke et al. (2005) and Sweller et al. (2011).

5.1.2 Method

In this study, students who were unfamiliar with Maple were introduced to it using one of two sets of instructional materials. The study required the participants to complete a sequence of instructional, testing, and workload assessment steps. In this section, details of the participants, the sequence of tasks in the study, the instructional materials, and the evaluation of participant learning are discussed.

5.1.2.1 Participants

Eighteen first-year engineering students participated in this study. None of the participants reported using Maple in the past.

5.1.2.2 Instructional Materials

From the previous study discussed in section 3.3, a WDA and CTA were conducted for Maple use, and from those analyses, a set of rule-based behaviours for using Maple were developed. Those rules used are shown in Table 11. For this study, two sets of Maple instructional materials were developed to teach the Maple concepts based on the identified rules, one set using familiar and one set using unfamiliar mathematics as context. These instructional materials can be found in Appendix C and Appendix D respectively. The instructional material types defined the two conditions for this experiment: familiar and unfamiliar.

The choice of mathematical concepts was based on an examination of the participants' curriculum for their current term, with the familiar math coming from early in the syllabus for the term and the unfamiliar math coming from late in the syllabus. The mathematical topics used for the familiar condition were finding the intersection of curves and finding a derivative and tangent line. In the unfamiliar condition, the instructional material covered approximation of a function with a Taylor series.

The participants in this study were assigned randomly to one of the two conditions. Each participant was introduced to a common set of Maple commands using one of the two different sets of instructional material, depending on their condition assignment.

It was hypothesized that the familiar mathematical examples would impose a lower extraneous cognitive load on the participants. As a result of not needing to expend cognitive resources on understanding the new mathematics, the participants in the familiar condition should have had more cognitive resources free for learning the Maple commands and concepts. The participants' effectiveness of learning of the Maple commands was measured using a test.

5.1.2.3 Evaluation of Participant Performance

After working through the instructional material, the participants were given a test to determine how well they had learned the rules. The participants were given the same test regardless of the condition they were assigned to. The test focused on the Maple rules and only used math that all the participants would be familiar with. There were 14 questions in the test that tested a number of Maple concepts: syntax, command names, and how commands work together to complete small tasks. The test can be found in Appendix E. It was expected that the participants in the familiar math instruction condition would perform better on the test, as measured by a larger number of correct answers.

In addition to the test, the NASA-TLX was administered to determine how much workload the participants experienced during the task of following the instructional material and completing the test. It was hypothesized that the participants in the familiar condition would score lower on the NASA-TLX after the test, indicating they experienced less workload than the participants in the unfamiliar condition did.

5.1.2.4 Experiment Sequence

The participants each completed the following sequence of tasks during the study:

1. Each participant completed a worksheet through which several Maple concepts based on the identified rules were introduced through examples. After the concepts were introduced, the participants used the concepts to solve one or more mathematical problems. Eight of the participants learned from the worksheet that used familiar mathematics and seven learned from the worksheet that used unfamiliar mathematics.

2. After completing all the problems in the worksheet, each participant completed the NASA Task Load Index (TLX) (Hart and Staveland, 1988; Hart, 2006) to determine how demanding he or she found the task in step 1. The TLX was administered using the computer-based version found at http://humansystems.arc.nasa.gov/groups/TLX/computer.php (NASA, n.d.b.)
3. Each participant completed a test that tested his or her knowledge of the rules for the Maple concepts introduced in step 1. There were 14 questions on the test.
   The test questions were designed to measure the basic rules needed to use the Maple concepts. The test was the same for both instructional conditions, and the focus of the test questions was on the names, syntax, and results of Maple commands. Any mathematics used, when necessary for context, was chosen from mathematics expected to be familiar to all participants.
4. Each participant completed the NASA-TLX again to determine how difficult he or she found the test task in step 3.

5.1.3 Results

There were two types of results for the study: the Maple command test results and the NASA-TLX results. Three participants' data were excluded from the results. Participant 1 did not have NASA-TLX results and participant 3 had missing data points, due to missing video.

Participant 18's NASA-TLX results indicated that he found the study much easier than the other participants did. Not including participants 1 and 3, the mean (and standard deviation) for the NASA-TLX results for the instructional materials and test were 41.4 (18.04) and 47.20 (16.52). Participant 18 had results of 6.67 and 6.68 respectively. Therefore, participant 18's results have z-scores of 2.25 and 2.44. This experiment was designed to involve participants with little experience with the new mathematics and with Maple-type programming. Because of these extreme results that could indicate the participant 18 had more knowledge than the other participants, his results are also excluded. These exclusions left 8 participants in the familiar condition and 7 in the unfamiliar condition.

5.1.3.1 Test Results

There were 14 questions in the test. The results for all participants for the test results can be found in Appendix F.

The medians for the familiar and unfamiliar groups were 12 and 10 correct answers respectively. A Mann-Whitney U test on the data revealed a significant effect of instructional condition (U = 49.5, $n_{familiar} = 8, n_{unfamiliar} = 7$, p=0.012) on the number of correct answers.

5.1.3.2 NASA-TLX Results

As discussed in the methods section, the participants were asked to complete the NASA-TLX twice, to assess how difficult they found the instructional material and the test. The TLX results for all participants are found in Appendix G. A lower TLX score indicates a lower workload.

For the post-instructional material NASA-TLX, the median values of the familiar and unfamiliar conditions were 48.95 and 42.3 respectively. A Mann-Whitney U test did not reveal a significant difference between instructional conditions (U = 32.5, $n_{familiar} = 8, n_{unfamiliar} = 7,$ p = 0.643). For the post-test NASA-TLX, the median values of the familiar and unfamiliar conditions were 45.5 and 55.8. A Mann-Whitney U test indicated a significant effect of instructional condition on the post-test NASA-TLX (U = 10, $n_{familiar} = 8, n_{unfamiliar} = 7,$ p = 0.040).

5.1.4 Discussion

The results show that the participants in the familiar math condition did better on the test and that they found the test easier, based on the test and NASA-TLX results. These results support the hypothesis that the participants in the familiar math condition had learned more of the rules for the use of Maple, indicated by those participants completing more test questions correctly and with less workload.

The lack of significant difference in NASA-TLX scores for the instructional materials is explainable by referring back to CLT. Participants would have the same amount of cognitive resources to use in completing both sets of instructional materials, but the design of the materials could cause the participants to use those resources differently. In the case of the unfamiliar condition, the resources would be used to process and form schemas relevant to two sets of concepts, Maple and mathematics, while in the familiar condition, the same cognitive resources could be focused more intensely on forming schemas germane to Maple commands and concepts. These results are consistent with the hypothesis that participants in the familiar math condition would learn the rules for Maple commands better, because they did not have to use cognitive resources on the learning of mathematics.

The Maple commands that the participants learned in this study represent only a portion of the knowledge base of rules that might be needed to use Maple for math problems that the participants might encounter, but the rules that were taught to the participants were developed in accordance with the three guidelines presented in Section 4.1 for placing an operator on a learning trajectory. In other words, the rules that the participants learned form a basis, in theory, for being able to interpret future math problems in the form needed to use Maple to assist with a solution. Therefore, by learning the Maple rule-based behaviours better, the participants in the familiar math condition should be better able to interface with the Maple system in the future. Such a prediction is consistent with the finding of Clarke et al. (2005) that pre-training of a tool helps with future use of that tool for learning of new mathematics. However, future work could test if the rules taught are of use in new situations and if those participants who learned the rules with the lower cognitive load (familiar) learning material would be better able to transfer the rules.

5.1.5 Limitations

There are a number of limitations on the conclusions that can be drawn in this study. The first of these problems extends from the limitations that ensuring ecological validity sometimes places on other types of validity in a study (Proctor & Van Zandt, 2008). This study has not shown for certain that the difference in mathematical material in the instructional materials caused a change in cognitive load that cause the difference in test results; however, this limitation is consistent with previously mentioned EID studies (Christoffersen et al., 1996; Vicente et al., 2005), where the intended manipulations were not the only possible change between conditions.

Furthermore, in this study, no measurement was made of the participants' prior programming or mathematical knowledge beyond asking the participants if they had previous experience with Maple. Therefore, it is not certain how uniform the pre-existing knowledge of the students was as related to programming or mathematics. Additionally, this lack of information about pre-existing knowledge made it difficult to assess why participant 18 performed much better in the study than the other participants.

As was mentioned, this study examined how many rules the participants learned. However, in a future study, it would be useful to test if the participants in the familiar condition were better prepared to solve more complex problems. Such a result would be predicted by the theoretical work in this dissertation and consistent with the previously mentioned work of Clarke et al. (2005), Schnotz and Kürschner (2007), and Sweller et al. (2011).

Finally, The NASA-TLX procedure used in this study did not involve any measurement of workload on a reference task, so individual differences could not be removed from the results. In the next study there was a task common to the two conditions, and, therefore, individual differences could be partially accounted for.

5.1.6 Summary

The goal of this study was to test the hypothesis that students who are trying to learn rules for using a system, such as Maple, would learn those rules better if their cognitive resources were focused on that learning task, as opposed to engaged in high levels of cognitive load induced by new contextual information. Specifically, a test of the hypothesis, generated during the field study in section 3.1, that students would learn Maple command- and syntax- related rules better if the instructional material used mathematics with which the participants were familiar was performed. The results of this study support the hypothesis that the students in the familiar condition learned more of the rules related to Maple commands and concepts, indicating those students had a better knowledge base of the concepts identified in the analysis of Maple-related rules. This knowledge base could then, in theory, be used to complete more complex problems and learn new mathematics with the assistance of Maple, a hypothesis worth testing in a future study.

The current study tested the learning of rules under differing conditions of cognitive load. However, effective training must support operators in novel situations. Therefore, the next study looks at how students transfer rule-based behaviour from learning tasks to novel testing tasks that could be performed more efficiently using the learned rules.

Chapter 6
Instructional Design and Learned Rule Transfer

The purpose of the previous study was to examine students' learning of rules by testing their knowledge of the rules after they completed one of two sets of instructional materials that imposed different levels of cognitive load. The students who used the instructional material designed to impose a lower cognitive load performed better on a test of the rules taught. However, that study did not examine how successfully the students used those rules during problem-solving transfer tasks. Understanding if the learning of rules will transfer to new tasks is important in assessing the benefit of the proposed training approach for problem solving. The study in this chapter examines the effect of cognitive load on the transfer of learning from instructional materials developed from a set of CWA-based rules. In particular, the study examines if two sets of instructional materials for teaching rules based on the guidelines in Chapter 4, and that are again differentiated using CLT concepts, result in different levels of performance on learning and transfer tasks.

In this study, a different problem-solving environment, Logo, was used. The use of Logo provided several benefits, as will be discussed. For this study, a set of rules, based on the guidelines in Chapter 4, was developed by analysing a WDA and CTA for a scaled-down version of Logo. Then the rules developed were used in the design of two sets of instructional materials, which are differentiated by cognitive load by using the CLT-based design recommendations described earlier and defined the two conditions for this study: high and low cognitive load. The participants in the study followed a sequence of instruction, testing, and workload assessment steps. It was expected that the participants in the low cognitive load condition would learn the instructional materials more quickly and with less effort, as measured using number of problems solved correctly, completion times, and NASA-TLX results.

This chapter is organized as follows:

- Section 6.1 presents the general hypotheses for the study.
- Section 6.2 presents the method used in this experiment. The section describes the reasons for the choice of environment, a WDA and CTA for Logo, the derivation of the instructional materials used, the testing methods used, and the experiment sequence.

- Section 6.3 presents the results for the study including the time taken for each section and problem in the study and the corresponding success rates, along with the results of the analysis of strategies used for some of the problems in the study. All the significant time results are summarized in Table 76.
- Section 6.4 presents the study results and their implications.
- Section 6.5 discusses the limitations of the study.

6.1 Hypothesis

As in the previous study, it was hypothesized that the participants in the low cognitive load condition would learn rules more efficiently as measured through time and workload measures during instruction. Further, it was expected that because those participants would have learned more rules, they would be able to complete the tests more quickly and complete more test problems correctly, and demonstrate more instances of transfer of the rules identified for training.

6.2 Method

This study required the participants to complete a sequence of experience assessment, instruction, testing, and workload assessment tasks. In building up to the description of the experiment sequence, this section describes the choice of the problem-solving environment and then the development of the instructional and testing materials, including the WDA and CTA used in the derivation of a set of rules that guided the design of the instructional materials.

6.2.1 Participants

Forty-eight students from the undergraduate and graduate populations participated in this study. None of the participants in this study participated in any of the other studies described in this dissertation.

6.2.2 Why was Logo Used

There are a number of reasons why Logo was a useful problem-solving environment for this study including the ease with which it can be taught, the amount of feedback given to users, and the amount of data that can be collected.

First, Logo is a simple language, the basics of which can be taught to a person in a short period of time, but even with a short amount of instruction, a user can solve interesting problems (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko, & Miller, 1997; Boulay, O'Shea, & Monk, J, 1981). However, Logo still fits into the genre of problem solving technologies appropriate for an undergraduate audience (Harvey, 1997). Feurzeig, Papert, and Lawler (2011) in a reprint of Feurzeig and Papert's (1968) original paper on Logo, stated that Logo was designed to help teach students mathematics, including both numerical and non-numerical mathematics. Feuzeig (2010) noted that Logo has been used in the teaching of algebra and geometry, and it has also been used to teach computer science at the undergraduate level (see for example Harvey, 1997).

Second, the Logo language, particularly the iPad interface used for this study, has some useful characteristics. The interface limits the number of types of actions a user can take and gives instant feedback every time a command is changed in a program. These two properties allowed the participants to experiment to solve problems when they were not certain of how to proceed, allowing the participants to employ model-based behaviours in a search for problem solutions and possibly discover relevant rules for solving the problems, a learning path described by Rasmussen (1986).

Third, the implementation of Logo used allows the gathering of data describing every action taken when drawing a figure, which can be processed and analysed semi-automatically. The data gathered from Logo allowed for the detailed study of strategies that the participants used. This automatic gathering of detailed data would not have been possible with Maple.

6.2.3 Logo Interface

This study involved the participants using a scaled-down version of Logo, which uses a subset of commands normally found in Logo (Harvey, 1997). A sample screen from the iPad version implemented for this study is shown in Figure 12. In Logo, images can be created by issuing a number of commands to the "turtle", an onscreen pen that can write on the screen, move without writing, or change orientation. As the turtle moves, it leaves a line behind itself showing the path it has taken (unless the pen is turned off, in which case the turtle moves without leaving a line). Using sequences of commands, the Logo user can create programs to draw shapes on the screen. The shapes drawn can be quite simple, or in the case of repeated moves and when using subprograms, be quite complex.

There are two types of screens in the user interface for the iPad version of Logo used in this study, the list of programs and the command list for each program. In Figure 12, a list of programs is shown on the left and the image resulting from the selected program on the right.

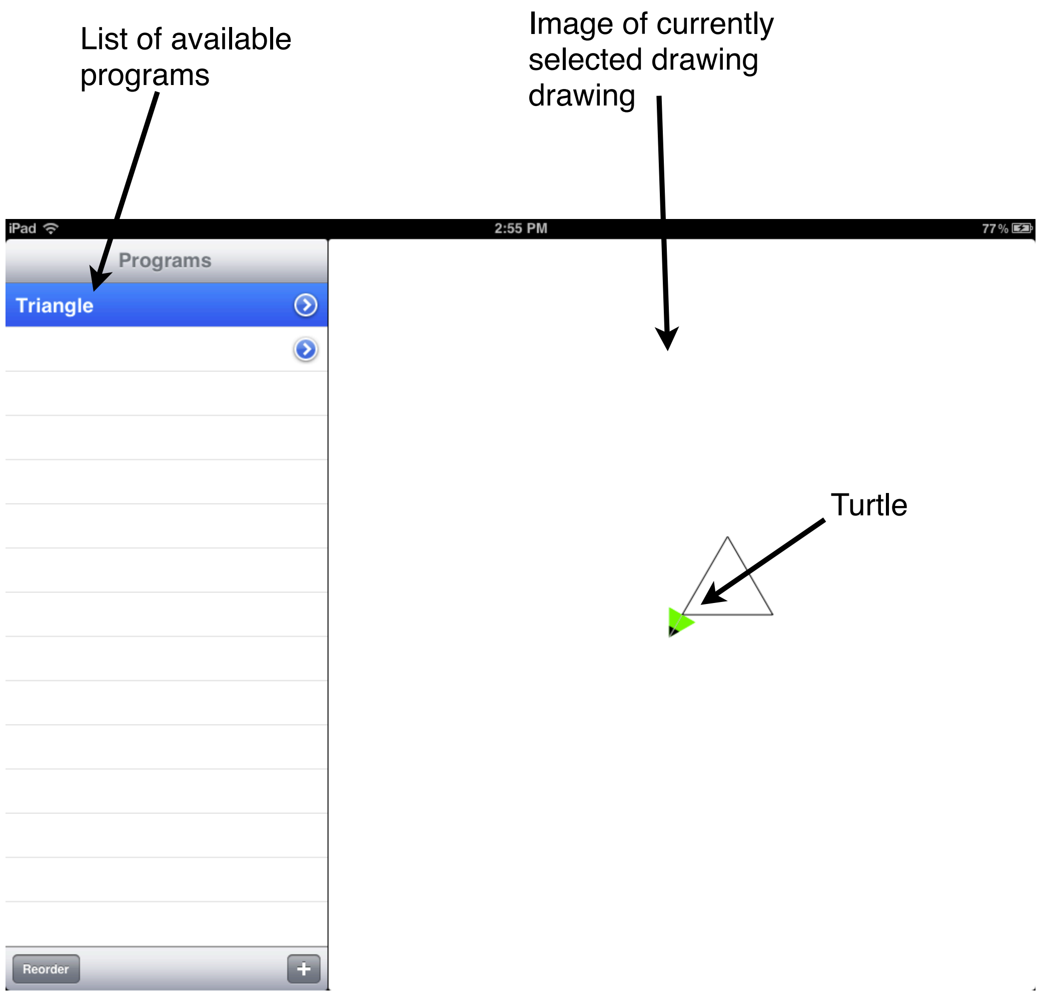
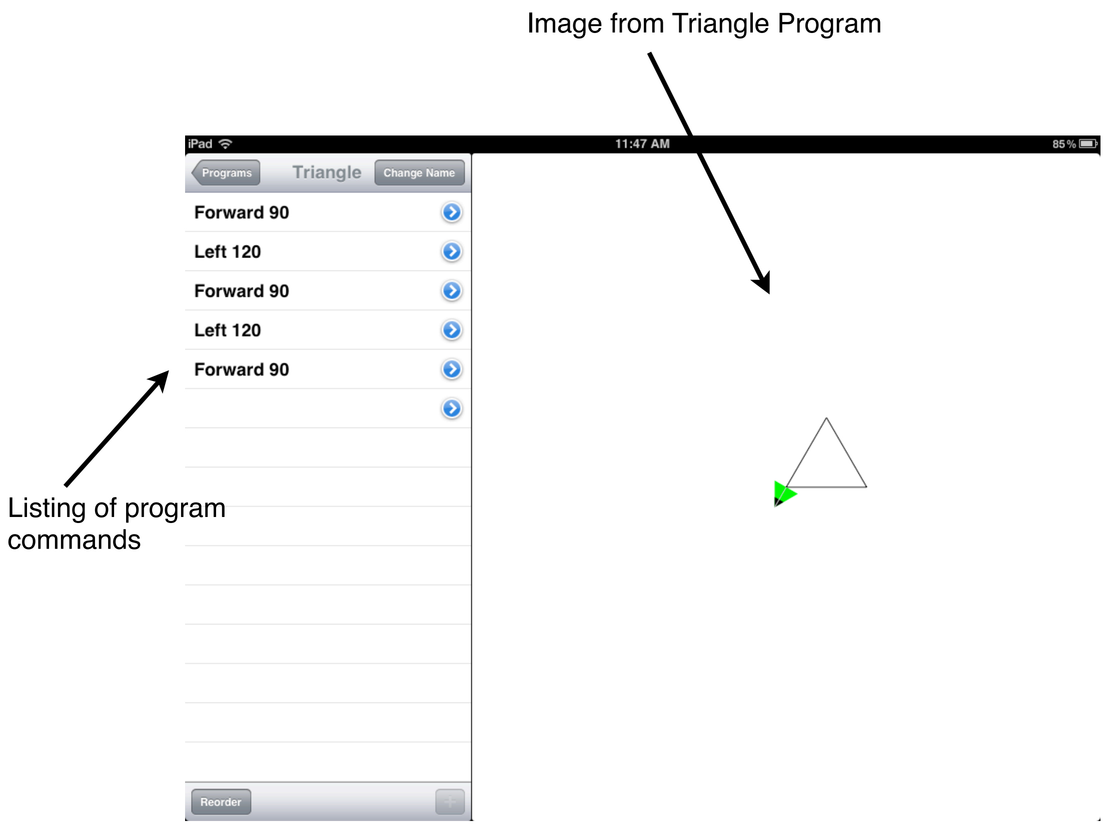

List of available programs
Image of currently selected drawing drawing
Turtle
iPad
2:55 PM
77%
Programs
Triangle
Reorder
+

Figure 12 - Sample Logo Screen from iPad

The program listing for the Triangle program is shown in Figure 13. On the left of Figure 13 is the list of commands that generates the image shown on the right. Each program consists of a number of commands, each of which is either a reference to another program or one of a number of primitive commands. The primitive commands are summarized in Table 12. Some of the primitive commands take a numerical parameter, N.



Forward 90
Left 120
Forward 90
Left 120
Forward 90

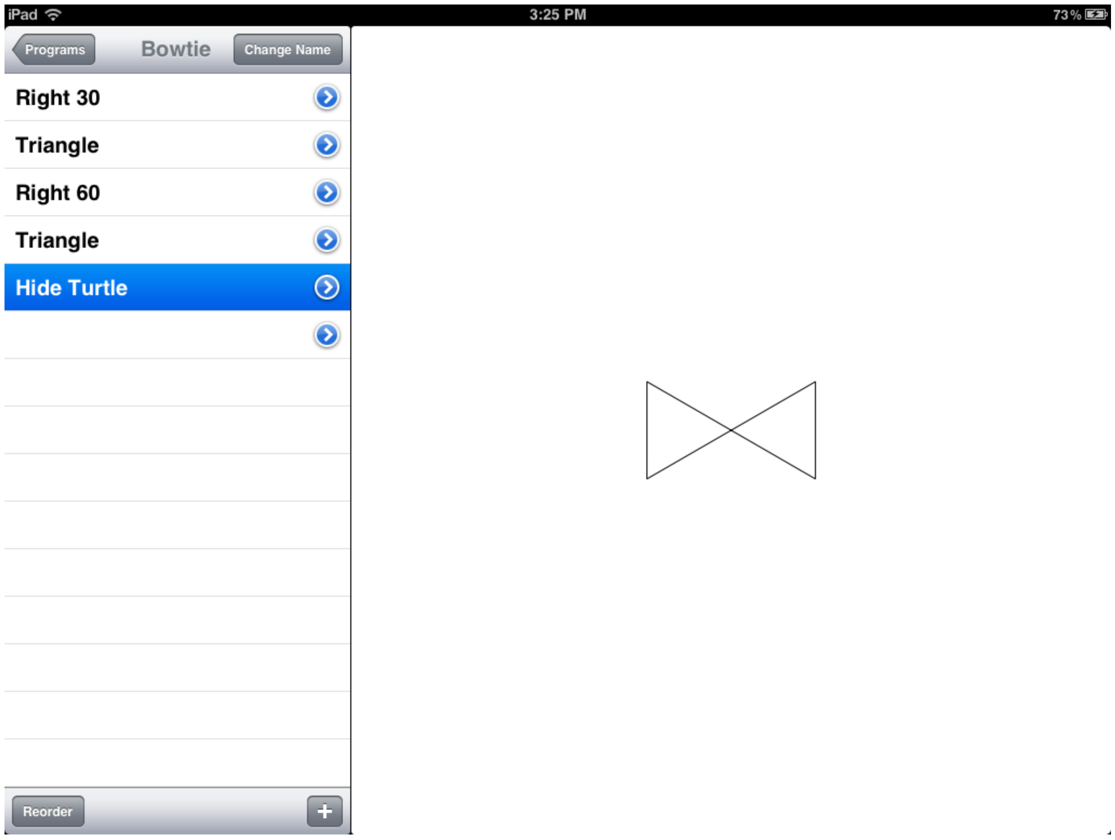
Figure 13 - Program listing for Triangle program

Table 12 - Primitive Logo commands

Command Name	Parameter	Use
Forward	N	Moves the turtle N pixels. If the pen is down, the turtle leaves a line as it moves.
Right	N	Turns the turtle N degrees to the right from its current heading.
Left	N	Turns the turtle N degrees to the left from its current heading.
Pen Up		Turns off drawing. Any subsequent forward commands will move the turtle but not draw any lines until a Pen Down command is issued.
Pen Down		Turns on drawing. Any subsequent Forward commands will move the turtle and draw a line until a Pen Up command is issued.
Hide Turtle/Show		Causes the turtle to be hidden or not. Only the last issues of these commands will have an effect.

Command Name	Parameter	Use
Turtle		
Repeat	N	Creates a loop. Any commands between the Repeat command and its associated End Repeat (which is inserted automatically) are repeated N times.

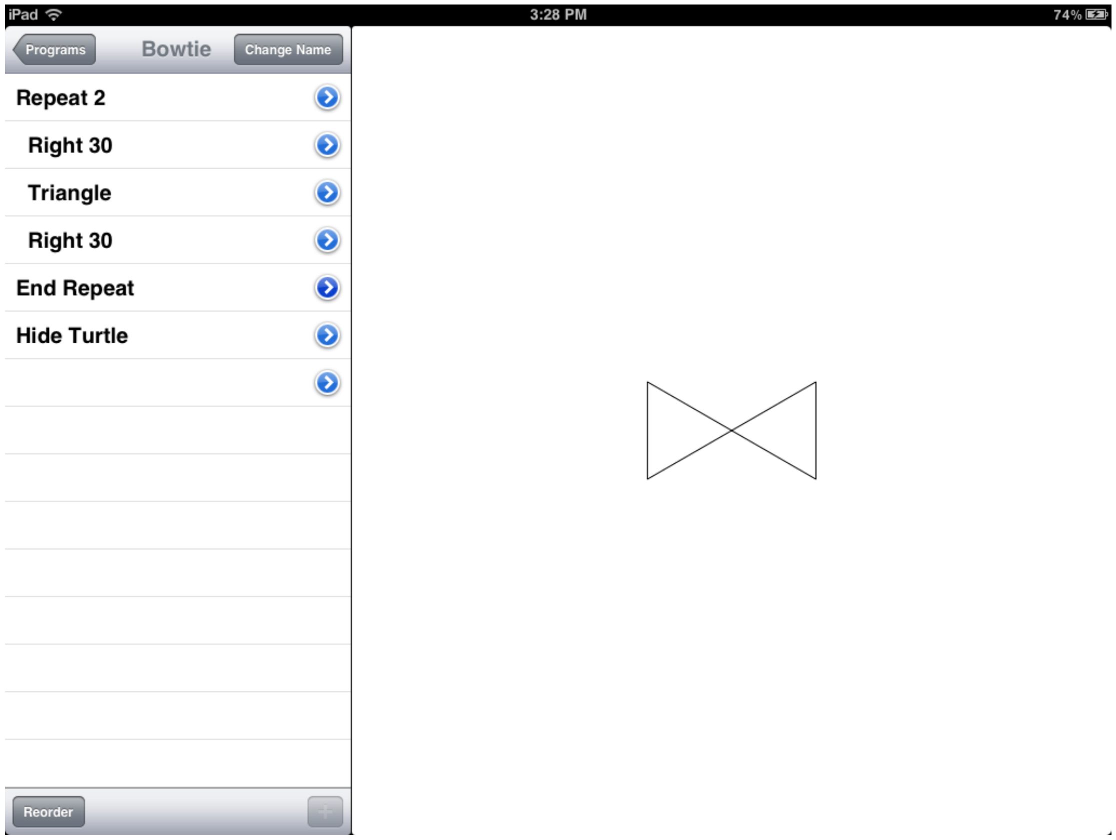
In addition to each of the above commands, the user can reuse other programs he or she has created as subprograms. Figure 14 shows a program called Bowtie that reuses the Triangle program twice. The use of subprograms allows the creation of more complex programs.



Right 30
Triangle
Right 60
Triangle
Hide Turtle

Figure 14 - Logo program that uses the Triangle subprogram

The Repeat command is a basic loop command, which allows the creation of more interesting programs. In Figure 15, the repeat command has been used to recreate the Bowtie.



```
Repeat 2
  Right 30
  Triangle
  Right 30
End Repeat
Hide Turtle
```

Figure 15 - Bow Tie program created using a repeat loop

This section has presented a basic introduction to the commands and interface for Logo. In the next sections the Logo language and its use is analysed with a WDA and CTA.

6.2.4 Logo Work Domain Analysis

From the design of the Logo program used for this study, an abstraction hierarchy was created, as shown in Figure 16. Logo, like Maple, is a programming language, so the abstraction hierarchies for the two systems are similar. In the case of Logo, the functional purpose is to draw certain figures. As shown in the abstract function, the drawing of those figures is guided by the principles of "turtle geometry" (Abelson, 1981). The general function level shows the processes of input, drawing algorithms, and output, which, together, result in the functioning of the system to produce the figures. All of those processes are achieved with the Logo command capabilities represented at the physical function level, and examples of which are given. The command capabilities used in a program are specified by a user with Logo command syntax, found at the physical form level. The command syntax can be broken into a command name, examples of which are listed, and, sometimes, an amount.

As a user creates a figure, he or she would have to change the state of the work domain represented in Figure 16. How a user can modify the state of the work domain can be described by a decision ladder.

6.2.5 Decision Ladder for Logo

The standard task of completing a figure in Logo by changing the state of the work domain can be considered using a CTA decision ladder. Recall that the decision ladder is related to the WDA abstraction hierarchy because the decision ladder describes possible tasks that can be completed to modify the state of the system described in the work domain to accomplish goals. The decision ladder can then be used to consider training needs and identify potential cognitive shortcuts.

Figure 17 shows a possible decision ladder for a task of completing a figure in Logo. Such a task would involve the manipulation of the turtle to change a current figure, possibly a blank figure, to match a goal figure. Table 13 describes the various states of knowledge in the decision ladder. The table gives the descriptions of each state as relevant to Logo.

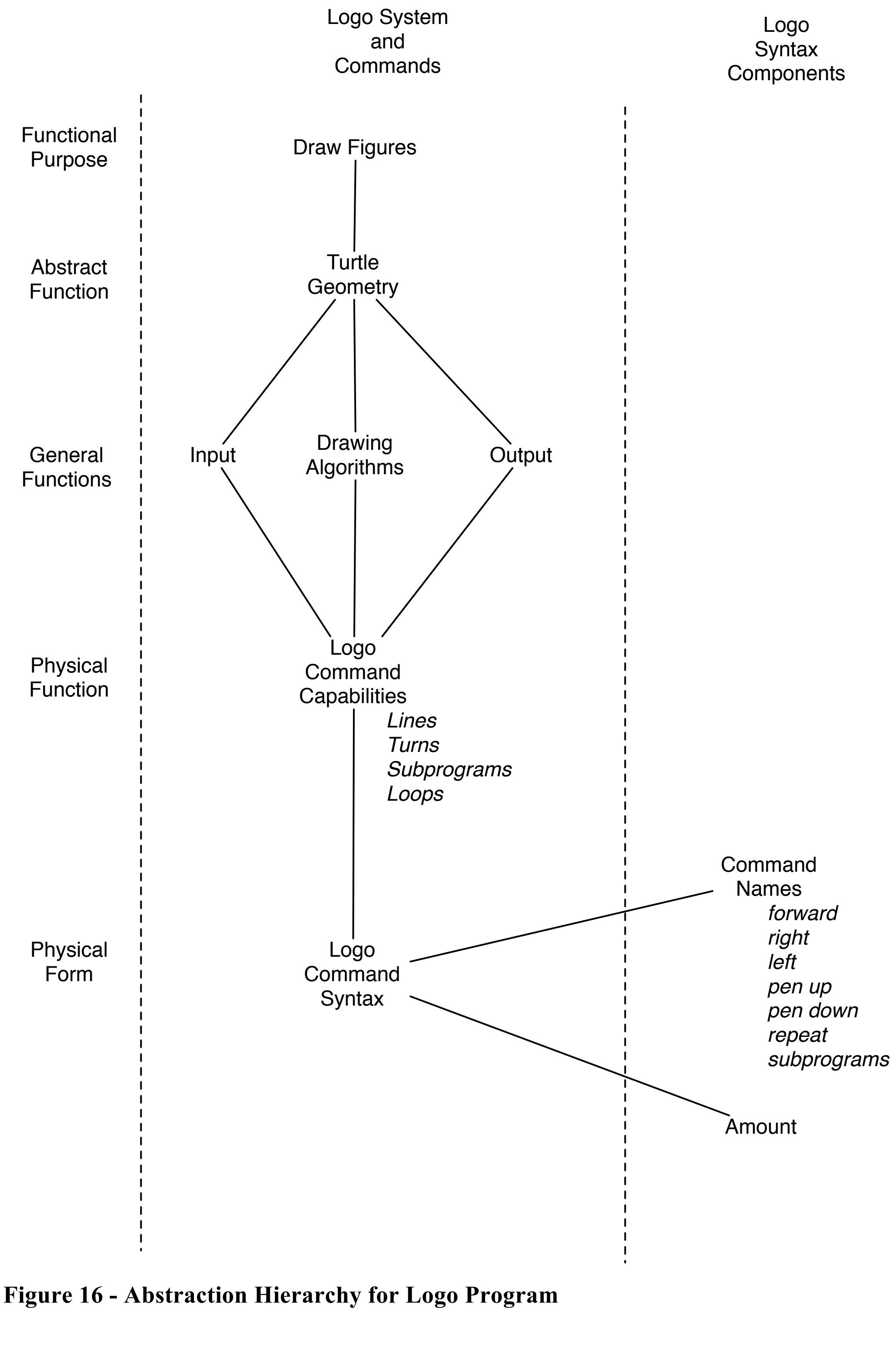


Figure 16 - Abstraction Hierarchy for Logo Program

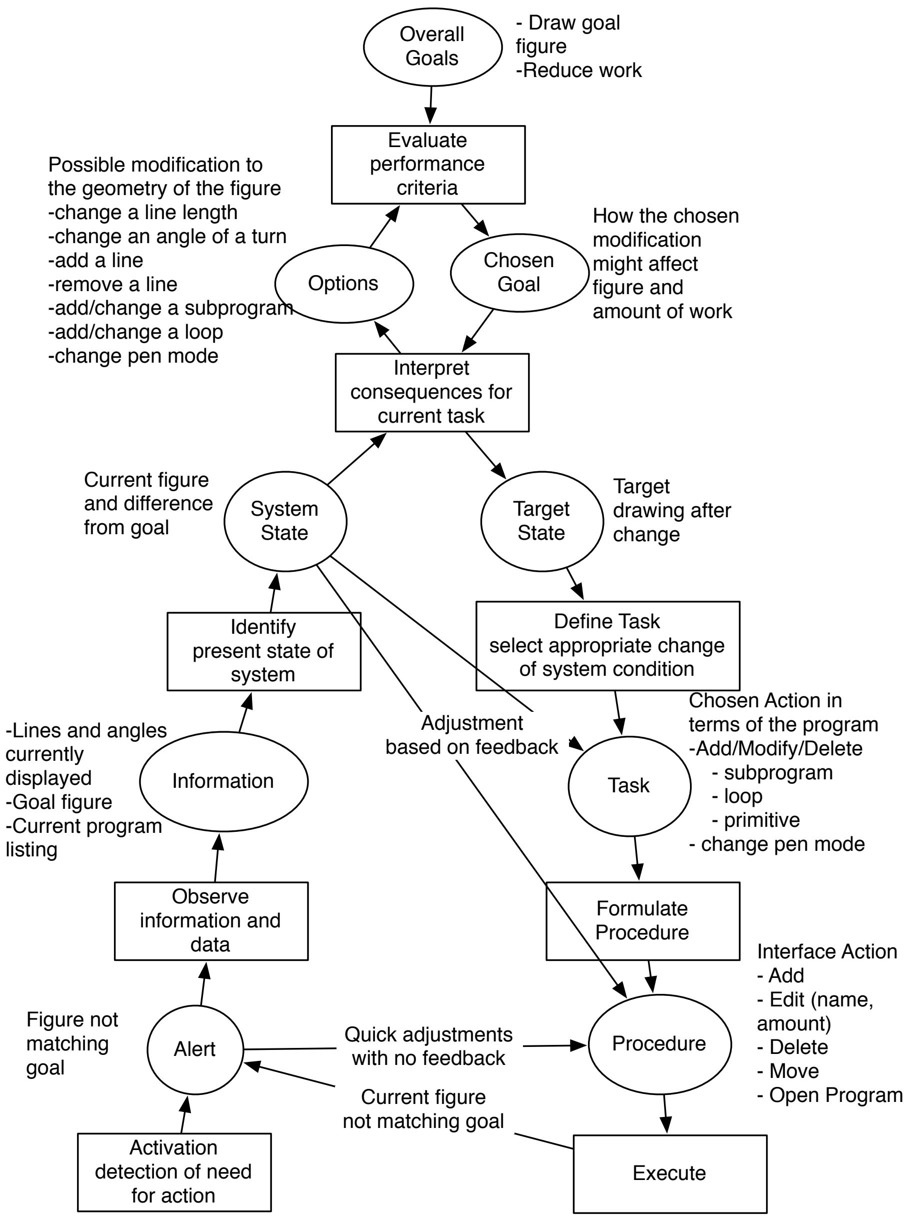

Overall Goals
- Draw goal figure
-Reduce work
Evaluate performance criteria
Possible modification to the geometry of the figure
-change a line length
-change an angle of a turn
-add a line
-remove a line
-add/change a subprogram
-add/change a loop
-change pen mode
Options
Chosen Goal
How the chosen modification might affect figure and amount of work
Interpret consequences for current task
Current figure and difference from goal
System State
Target State
Target drawing after change
Identify present state of system
Define Task select appropriate change of system condition
Adjustment based on feedback
Chosen Action in terms of the program
-Add/Modify/Delete
- subprogram
- loop
- primitive
- change pen mode
-Lines and angles currently displayed
-Goal figure
-Current program listing
Information
Task
Observe information and data
Formulate Procedure
Interface Action
- Add
- Edit (name, amount)
- Delete
- Move
- Open Program
Figure not matching goal
Alert
Quick adjustments with no feedback
Procedure
Current figure not matching goal
Activation detection of need for action
Execute

Figure 17 - Decision ladder for drawing a figure in Logo

Table 13 - Descriptions of states in Logo decision ladder

State	Purpose in analysis
Alert	The user detects that the figure he or she currently has drawn does not match the goal figure.
Information	The user has access to information such as the goal figure, the lines and angles that constitute the current figure, and the current program listing. These pieces of information may help the user decide what action to take.
System State	The current figure and the difference between the current figure and the goal figure.
Options	There are a number of possible modifications that the user could make to the geometry of the figure including adding, removing, or changing a line, angle, subprogram, loop, or pen-mode adjustment.
Overall Goals	The overall goal identified in the WDA is to draw the figures given. The user may also want to reduce the amount of work they need to do in completing a problem.
Chosen Goal	Once an option is chosen, the user could consider how that option will affect the goal, including how the option will affect the figure as defined by turtle geometry.
Target State	The target figure after taking an action.
Task	The task represents a change to the program listing representing an action that will achieve the target figure.
Procedure	Once a task is chosen, an appropriate interface action must be selected to implement the task such as changing a command, opening a program, and a number of other possible actions.

The decision ladder presented in Figure 17 represents a map of possible cognitive states and processes. The complete decision ladder represents a formal decision-making process, meaning that a user using Logo may explicitly consider the state of the system, what options are available, how the options affect the drawing, and then how to implement the option chosen. For example, a user might explicitly consider possible subprograms and how they might fit into a figure he or she is trying to draw. However, as discussed, real decision-making often involves shortcuts driven by rules (Rasmussen, 1986; Lintern and Naikar, 1998). For example, a user, after gaining some experience, might recognize a subprogram's figure within a larger figure, and leap from system state to a task of using a particular subprogram.

Furthermore, it is possible, and quite likely, that users will make errors during the various processes described by the decision ladder. For example, a user could misinterpret or misunderstand how a chosen action might affect the figure or not map their desired option to the proper task for a change in their program. When a user makes an error in some task described in the decision ladder, he or she can use the feedback from the Logo system to recognize that his or her current figure is still not matching the goal figure and return to the alert state and from there continue to make adjustments.

In addition to returning to the alert state to correct errors, a user may make other shortcuts in the decision ladder to reduce workload. For example, for some of the problems discussed later, a user may need to fine tune angles in their figure. In such cases, he or she might cycle through the alert-procedure-execute cycle shown when making adjustments to angles, without considering the rest of the decision ladder states. In other situations, the user might need to consider some feedback from Logo and use a particular task or procedure to make an adjustment. Because rule-based behaviours are generally less cognitively demanding than model-based behaviour (Rasmussen, 1986), if a user learns to make rule-based shortcuts in the decision ladder, he or she may be able to complete problems more quickly and with less workload. The total collection of the choices and shortcuts that a user makes, as described by the decision ladder, forms the strategies that he of she can use in drawing a figure.

Given the shortcuts a user could make, and the impossibility of always knowing what path a person will take through the decision ladder (Rasmussen, 1974), the decision ladder is not an exact description of a cognitive process (as noted in Vicente, 1999). However, as a map of cognition, the decision ladder, along with the abstraction hierarchy, can still be used to describe and derive rules used to develop training materials.

6.2.6 Instructional Material

There were three different stages of instruction in the study: introduction to Logo, subprograms, and loops. The development of the instructional materials, based on rules derived from the WDA and CTA, will be described; however, first, a high level overview of the instructional materials will be given.

All participants followed the same instructional material, found in Appendix I, for the introduction to basic Logo concepts including use of the interface and the basic, built-in Logo commands. The participants were then introduced to more advanced Logo features through one of two sets of instructions that defined the high and low cognitive load conditions in this study. The instructions for the different conditions used different types of instructional material to introduce them to two key Logo concepts: subprograms and loops.

The two sets of instructional materials used the same set of problems but were designed to impose one of two levels of cognitive load on each participant, based on which of the two conditions a participant was randomly assigned to. The instructional material used by participants in the low cognitive load condition was designed to reduce cognitive load for the problems through the different methods of controlling cognitive load discussed in section 2.7.4. As a reminder, these methods include worked examples and completion problems. The instructional material for the high cognitive load group used traditional, unaided problem solving for most problems; however, for some problems, worked examples and completion problems were still used to introduce new concepts to the participants.

6.2.6.1 Introductory Material

As discussed, all participants in the study completed the same introductory materials to introduce them to the basic logo commands and interface use. Again, the introductory material used can be found in Appendix I.

Table 14 shows the concepts introduced in the introductory material that could be used as rules in the future, how the concepts are related to the abstraction hierarchy and decision ladder in Figure 16 and Figure 17 respectively, and how the potential rules are related to the three guidelines from section 4.1. As in Elix and Naikar (2008), the relation of the training material to the decision ladder is described via the knowledge states. The first few concepts and rules are related to the interface for logo, without concern for the relation to programs and figures, but in the later examples, basic commands, and their effect on figures, are introduced.

Table 14 – Concepts and rules in introductory material

Concept or potential rule	Relation to AH and *DL (in italics)*	Recognize Signs	Appropriate Context	Chaining and Generalizing
Viewing a program	*Procedure*	Recognize the need to open a program and how to do so	Realize the need to be at the program list screen to open a new program.	
Rename a program	*Procedure*	Know how to change a program name	Recognize the need to open program open to change its name	
Delete a command	*Procedure*	Know how to delete a command		Future program work will probably require deleting commands. Can use concept as part of future program writing.
Add a command	*Procedure*	Know how to add a command including setting the command name and amount parameter		Future program work will require adding commands. Can use concept as part of future programs writing.
Move a command	*Procedure*	Know the sequence of actions to move a command	Recognize the need to switch to the reorder mode to move a command	Future program work will probably require moving commands

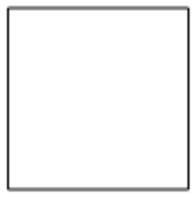
Concept or potential rule	Relation to AH and *DL (in italics)*	Recognize Signs	Appropriate Context	Chaining and Generalizing
Forward Command	Command Capabilities Command Syntax *Options* *Chosen Goal* *Target State* *Task*	Know how to move the turtle forward to draw a line		Future programs will involve drawing many lines, so need to know the effect of doing so. Can use concept as part of future programs.
Turning Commands (Right, Left)	Command Capabilities Command Syntax *Options* *Chosen Goal* *Target State* *Task*	Know how to turn the turtle when needed.		Future programs will involve drawing many turns, so need to know the effect of using turns. Can use concept as part of future programs.
Pen Up and Pen Down	Command Capabilities Command Syntax *Options* *Chosen Goal* *Target State* *Task*	Know how to turn the pen on and off		Future programs could involve moving the turtle without drawing, so this rule could be useful in such circumstances.

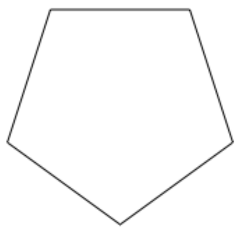
After completing the introductory material, the participants were introduced to more advanced Logo features through one of two sets of instructions that defined the high and low cognitive load conditions in this study. The instructions for the different conditions use different types of instructional material to introduce them to two key Logo concepts: subprograms and loops.

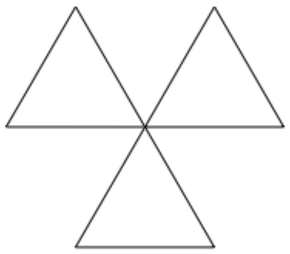
6.2.6.2 Subprograms Instruction

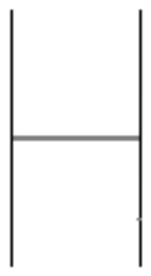
In the subprograms instruction, the participants were taught to draw basic figures and use subprograms to draw more complex figures. Table 15 presents each of the figures the participants were asked to draw, an image of the final figure, the concepts or rules taught, the relation of the concepts or rules to the abstraction hierarchy (referred to as AH in the table) in Figure 16 and the decision ladder (referred to as DL in the table) in Figure 17, and how the rules might relate to the three guidelines from section 4.1 (sign recognition and action, appropriate context, and chaining or generalizing). Additionally, the types of cognitive supports in the two types of instructional materials are summarized.

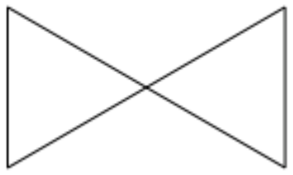
Table 15 – Problems used in subprogram instruction

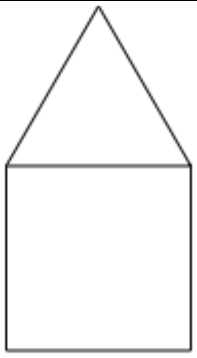
Problem	Final Figure	
Square		Concepts/Rules: Draw a simple shape from scratch using primitive components. Relation to AH: Command Capabilities Relation to DL: Target State (what the results of actions will be) Recognize/Action: Recognize when the primitives practiced can be useful and their effect. Appropriate Context: Chaining/Generalizing: Low Cognitive Load: Worked example (the participants were given the completed problem) High Cognitive Load: Traditional problem (the participants were given the line lengths told to draw the shape given)

Problem	Final Figure	
Pentagon		Concepts/Rules: Choice of angles for a given shape. Relation to AH: Command Capabilities Relation to DL: Options, Interpret, Target State (effect of choice of angles) Recognize/Action: Recognize the use of primitives needed for turns. Appropriate Context: Recognize what angles are needed for certain shapes. Chaining/Generalizing: Low Cognitive Load: Worked Example (given exact commands to add) High Cognitive Load: Traditional problem (given a word description of the figure to draw)

Problem	Final Figure	
Radiation		Concepts/Rules: Use of subprograms to draw more complex figures Relation to AH: Command Capabilities Relation to DL: Options and Chosen Goal (how the choice to use a subprogram affects drawing figure and work) Recognize/Action: Recognize when subprograms can be used Appropriate Context: Chaining/Generalizing: Future programs will involve the use of subprograms. Operator can start to generalize when subprograms will be useful. Low Cognitive Load: Worked example (participants given the completed problem using Triangle 3 times) High Cognitive Load: Completion problem (participants shown to use the Triangle program to draw one blade then told to complete the figure given)

Problem	Final Figure	
H		Concepts/Rules: More practice with use of primitives Relation to AH: Command Capabilities Relation to DL: Options, Target State, Task Recognize/Action: Recognizing what turtle moves are needed Appropriate Context: Chaining/Generalizing: Low Cognitive Load: Completion problem (participants given program for a nearly completed H that needed a few adjustments) High Cognitive Load: Traditional problem (participants given final image and line lengths)

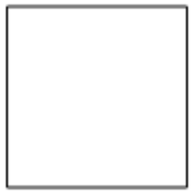
Problem	Final Figure	
Bow Tie		Concepts/Rules: Combination of subprograms with other commands. Learn to adjust angles between subprograms. Relation to AH: Command Capabilities and Syntax (particularly subprograms) Relation to DL: Participants may make multiple trips around the decision ladder, combining different types of options, or possibly make shortcuts while adjusting angles. For example, a participant might iterate between alert and procedure, without explicitly considering the meaning of their actions, adjusting an angle until the two triangles are properly oriented. Options and Chosen Goal (how the choice to use a subprogram affects drawing figure and work) Alert and Procedure Recognize/Action: Recognize when subprograms can be used. Appropriate Context: Chaining/Generalizing: Start to see when subprograms can be used as part of a program and this knowledge may help with future, more complex problems. Learn to adjust angles as part of creating a program. Low Cognitive Load: Completion Problem (given steps to complete) High Cognitive Load: Traditional Problem (given final figure and line lengths)

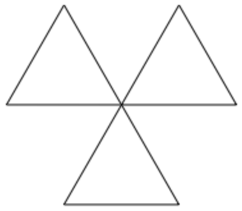
Problem	Final Figure	
House		Concepts/Rules: Assembling multiple subprograms. Relation to AH: Command capabilities (particularly subprograms) Relation to DL: Options, Chosen Goal, Target State (how the choice to use a subprogram affects drawing figure and work) Recognize/Action: Recognize when subprograms can be used Appropriate Context: The location of the turtle in at the end of the program that forms a subprogram affects the drawing of a program in which the subprogram is used. Chaining/Generalizing: Low Cognitive Load: Traditional problem (participant given final figure and line lengths and suggestion to reuse existing programs) High Cognitive Load: Traditional problem (participant given final figure and line lengths and suggestion to reuse existing programs)

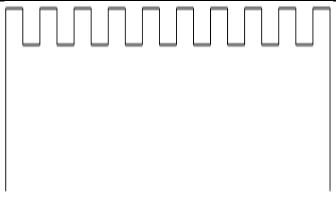
6.2.6.3 Loop Instructions

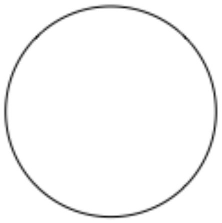
In addition to being taught to use subprograms, the participants were later taught to use loops. Similar to the subprograms instructions table above, Table 16 shows the problems used in the loop instructions.

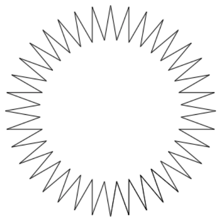
Table 16 – Problems used in loop instructions

Problem	Final Figure	
New Square		Concepts/Rules: Use of loops Relation to AH: Command Capabilities and Syntax Relation to DL: Choice of a loop in Options. The syntax for a loop in procedure. Recognize/Action: Recognize when loops can be useful Appropriate Context: Chaining/Generalizing: Start to generalize when loops will be useful Low Cognitive Load: Worked Example (participants given steps to complete problem) High Cognitive Load: Worked Example (participants given steps to complete problem)

Problem	Final Figure	
Radioactive Two		Concepts/Rules: Use of loops Relation to AH: Command Capabilities Relation to DL: Choice of a loop in Options and choice of amount in Options Recognize/Action: Recognize when loops can be useful Appropriate Context: Chaining/Generalizing: Continue to generalize when loops will be useful Low Cognitive Load: Worked Example (participants given steps to complete problem) High Cognitive Load: Worked Example (participants given steps to complete problem)

Problem	Final Figure	
Wall		Concepts/Rules: Use of loops combined with other primitives for drawing parts of the figure outside the loop Relation to AH: Command Capabilities Relation to DL: Options, Target State, Chosen Goal (when to use a loop and when not to use a loop but use other primitives) Recognize/Action: Recognize when a loop can be used to draw part of a figure Appropriate Context: Recognize when parts but not all of a figure can be drawn with a loop Chaining/Generalizing: Future problems will combine loops with other parts of figure outside the loops. Low Cognitive Load: Completion Problem (Given various incomplete steps) High Cognitive Load: Completion Problem (Given left line then asked to complete the problem)

Problem	Final Figure	
Circle		Concepts/Rules: Drawing curved shapes is done by repeatedly drawing a number of small lines interspersed with small turns Relation to AH: Command Capabilities (see that loops can be used to draw smooth-looking curves) Relation to DL: Options, Chosen Goal, Target State (how very small moves contribute to the overall goal of a particular figure) Recognize/Action: Recognize curved shapes and their relation to straight lines Appropriate Context: Chaining/Generalizing: Future problems could consist of curved sections so the idea of using many small moved for curved sections could be reused. Low Cognitive Load: Worked Example (given commands) High Cognitive Load: Worked Example (given commands)

Problem	Final Figure	
Starburst		Concepts/Rules: Learn to create a more complex shape in steps using a number of steps. Relation to AH: Command Capabilities (particularly the repeat command) Relation to DL: Will need to cycle through the decision ladder. Will need to consider the system state – the difference the current figure from the goal and options to get a little closer (particularly choice of angles between lines) Recognize/Action: Recognize when loops are needed to solve problems with repetitive components. Learn how to adjust angles between the lines between the line segments. Appropriate Context: Learn what angles are needed between the lines. Chaining/Generalizing: Recognize a partial solution and how it can lead to a final goal. Low Cognitive Load: Completion problem (given steps to create a line of zig-zags and then asked to complete the problem) High Cognitive Load: Traditional Problem See section 6.3.3.3 for details on this problem

There were a total of 11 problems in the instructional sets for subprograms and loops. In the high cognitive load instructions, traditional, unaided problem solving was used except where new concepts were introduced including subprograms, loops, use of loops for part of a program, and curved shapes. In the low cognitive load condition, a combination of worked examples and completion problems were used, except for one traditional problem, House, used for practice at the end of the subprograms instruction.

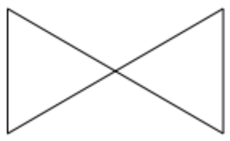
6.2.7 Testing Materials

After completing each of the subprogram and loop instructional material, the participants completed a test of the concepts taught in the instructional material. Some of the test questions were very similar, or even identical, to instructional questions, and some were less similar, requiring more distant transfer of the concepts learned.

The participants completed two tests during the study: one on subprograms and one on loops. All participants, regardless of condition, were given the same test. The tests were designed to measure how well the participants could reproduce and transfer the knowledge learned during the instructional steps. For each test problem, the participants were given the final image (except for one problem where a written description was given), and line lengths, and asked to draw the image. The tests can be found in Appendix L and Appendix M.

Table 17 and Table 18 show the problems that were in the tests. Each problem was associated with some of the problems in the instructional material in order to test the concepts taught. However, the problems in the tests were often more complicated than the problems in the instructional material, and, therefore, the test problems tested a mixture of concepts from several problems from the instructional material.

Table 17 – Problems used in subprograms test

Problem	Final Figure	
Bow Tie Two		Concepts/Rules: Use of subprograms and other commands. Related Instructional Problems: Bow Tie

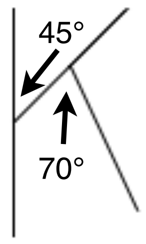
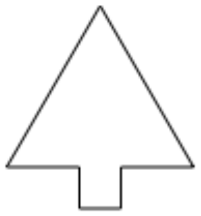
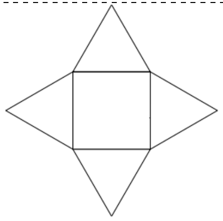
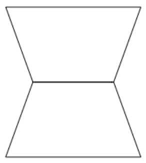
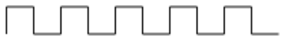
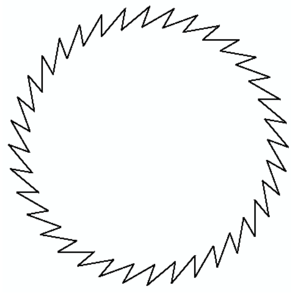
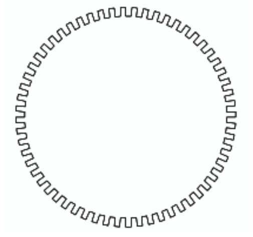
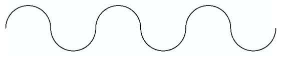
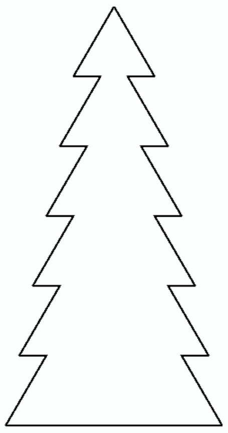
Problem	Final Figure	
K		Concepts/Rules: Use of primitives Related Instructional Problems: H
Up Arrow		Concepts/Rules: Draw a figure that is based on figures already seen. Related Instructional Problems: Triangle (from introductory material), Square
Star		Concepts/Rules: Use of subprograms and correctly placing the figures drawn by those subprograms. Related Instructional Problems: Radiation, House, Bow Tie
Blade		Concepts/Rules: Use of subprograms and correctly placing the figures drawn by those subprograms. In particular the effect of the location of the turtle in the subprogram. Related Instructional Problems: House

Table 18 – Problems used in loop test

Problem	Final Figure	
Hexagon	No image given. Participants asked to create a six-sided figure with each side length 50 and each angle a left turn of 60 degrees.	Concepts/Rules: Use of repeat command and correct angle selection. Related Instructional Problems: Pentagon, New Square
Bumps		Concepts/Rules: Use of repeat, choice of repeat count. Related Instructional Problems: Wall
Saw Blade		Concepts/Rules: adjusting angles between line segments. Related Instructional Problems: Starburst See section 6.3.3.3 for details on this problem
Gears		Concepts/Rules: Use small lines and turns to create a curved shape. Adjusting angles between line segments. Related Instructional Problems: Circle
Wave		Concepts/Rules: Use small lines and turns to create a curved shape. Related Instructional Problems: Circle

Problem	Final Figure	
Tree		Concepts/Rules: Use of loops combined with other primitives for drawing parts of the figure outside the loop. Related Instructional Problems: Wall

The tests tested the transfer of the knowledge that participants acquired from the instructional material. In addition to the tests of Logo knowledge, after each stage in the study, the participants completed the NASA-TLX to assess how much workload was placed on them.

6.2.8 NASA-TLX

The participants completed the NASA-TLX five times to assess their workload during the five sections of the study. They completed the NASA-TLX after each of the introduction, subprograms instruction, subprograms test, loop instruction, and loop test. During each administration of the test, the participants completed both the weighting and ranking phases of the NASA-TLX.

6.2.9 Data Collection

During the study, data was collected in three ways. First, the participants completed, on paper, the survey of pre-existing knowledge, as described in step one of the method. Second, every time the participants took an action in Logo, the action was recorded, including when the action occurred and all the data necessary such that the student's Logo session could be reproduced. Finally, the NASA-TLX data was collected using an iPad-based version of the NASA-TLX, which I implemented.

6.2.10 Experiment Sequence

The participants in the study followed a sequence of eleven steps as they completed the study. The path through the various steps is shown in Figure 18. Steps in the middle of Figure 18 were identical for the two conditions, but steps on the left or right were different for the low and high cognitive load conditions. Each of the steps is described below.

1. Each participant completed a questionnaire to collect basic demographic data and information about previous knowledge. The questionnaire can be found in Appendix H. On the questionnaire, the participants were asked to provide their age, sex, current program, year, and term. They were also asked how many programming courses they had taken in high school and university, if they had used Logo or LEGO Mindstorms before, and to assess their own programming experience on a scale from no experience to expert.

2. The participants followed the introductory material, found in Appendix I, through which they were introduced to the interface for the Logo program and the basic Logo commands. The participants were asked to complete all the steps in the instructional material using the iPad-based Logo program.

3. The participants completed the NASA-TLX to assess their workload while completing the instructional material.

4. The participants followed instructions on using subprograms. There were two types of instructional materials for this step, one for each of the low and high cognitive load groups. The two sets of instructional materials for the low and high cognitive load conditions are found in Appendix K and Appendix J respectively. There was no time limit given to the students for completing the instructional material.

5. The participants each completed the NASA-TLX a second time to assess their workload during the subprogram instruction.

6. The participants each completed a test asking them to create a number of figures, some involving subprograms. The test is found in Appendix L. There was no time limit given to the students for completing the test.

7. The participants again completed the NASA-TLX to assess their workload during the subprograms test.

8. The participants followed instructions on using loops. There were two types of instructional materials for this step, one for each of the low and high cognitive load groups. The two sets of instructional materials are found in Appendix K and Appendix J respectively. There was no time limit given to the students for completing the instructional material.

9. The participants each completed the NASA-TLX again to assess their workload during the loop instructions.

10. The participants each completed a test asking them to create a number of figures that involved loops. The test is found in Appendix M. There was no time limit given to the students for completing the test.

11. The participants each completed the NASA-TLX a final time to assess their workload during the loop test.

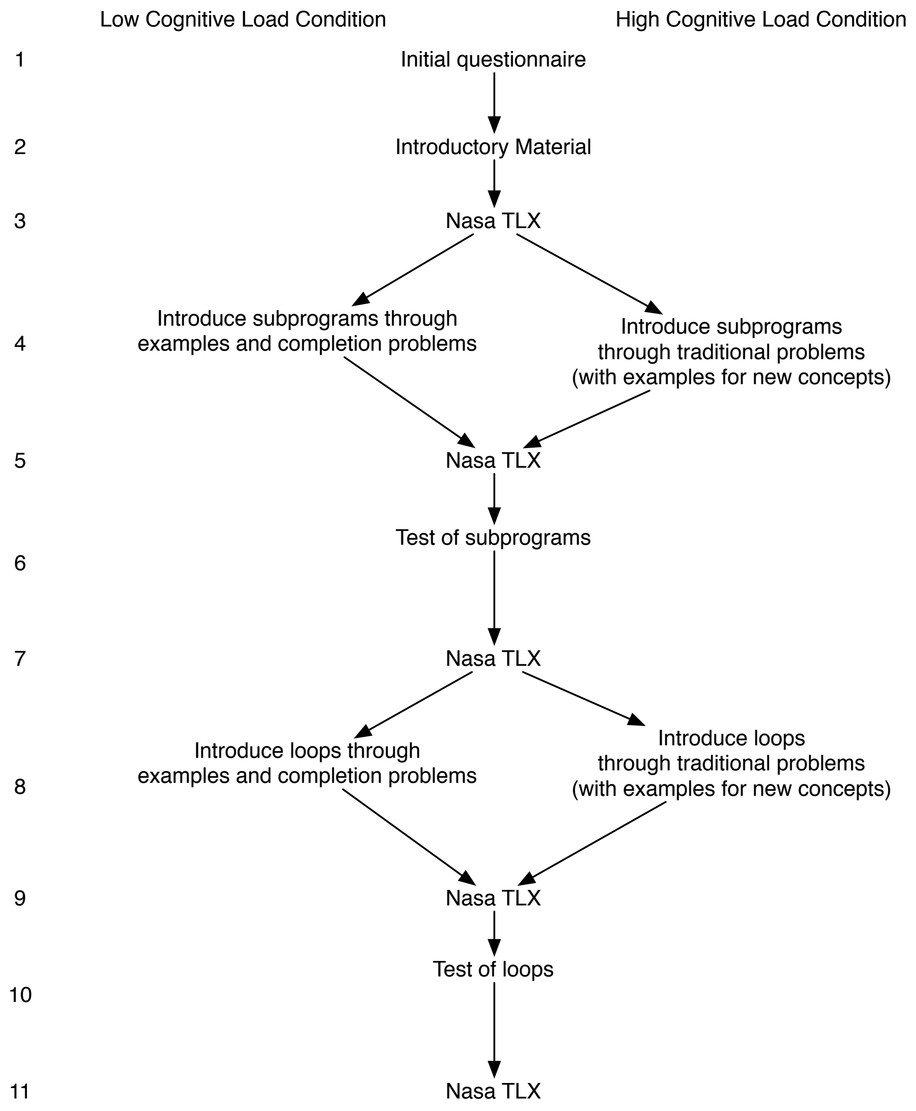

Low Cognitive Load Condition
High Cognitive Load Condition
1
Initial questionnaire
2
Introductory Material
3
Nasa TLX
4
Introduce subprograms through examples and completion problems
Introduce subprograms through traditional problems (with examples for new concepts)
5
Nasa TLX
6
Test of subprograms
7
Nasa TLX
8
Introduce loops through examples and completion problems
Introduce loops through traditional problems (with examples for new concepts)
9
Nasa TLX
10
Test of loops
11
Nasa TLX

Figure 18 - Sequence of steps for each participant in Logo study.

6.3 Results

In this section, all the results for the Logo study are presented. First, the results of the survey that assessed the participants' knowledge when each participant started the study are presented. Second, the timings and success rates for each section of the study are presented. Third, the results for each problem in the instructional and test steps are given. Fourth, detailed results are explored, including the use of certain strategies, for some specific problems. Finally, the NASA-TLX results are examined.

6.3.1 Participants, Survey Results, and Experience Levels

Forty eight participants completed the study. One participant did not follow the directions for the study and did not complete many of the problems. This participant's data was incomplete and could not be analysed, so has been excluded from the results.

The participants each completed the demographic survey found in Appendix H. The raw data from that survey are found in Appendix N.

Of the 47 remaining participants, 32 participants were male and 15 were female. The ages of the participants ranged from 18 to 31.

As discussed, the participants were asked to answer a questionnaire about their programming experience. They were asked to provide the number of computer science or programming courses taken in high school and university and assess their own programming experience. In the analysis of this study, one measure of experience, based on the participants' self-assessment, is used and in this section it is shown that all the measures of experience are related to that self-assessment. First, the data for each of the measures in the survey are summarized.

Table 19 and Table 20 show the number of participants who had taken various numbers of high school and university programming courses.

Table 19 - Number of participants who took a particular number of programming courses in high school

Number of Courses	0	1	2	3	4
Number of Participants	25	9	7	5	1

Table 20 - Number of participants who took a particular number of programming courses in university

Number of Courses	0	1	2	3	4	5	6	8	20
Number of Participants	21	11	6	2	3	1	1	1	1

As a measure of self-assessment of programing ability, the participants were asked to "Mark where you feel your programming experience lies on the following scale"

No Experience	Some Experience	High Level of Experience	Expert

Figure 19 – Logo experience survey self-assessment scale

The scale was scored as a value between 0 and 3 inclusive such that No Experience scored 0 and Expert scored 3. Any participant who marked the line but not at one of the values specified below the line had their score rounded to the nearest 0.5. Table 21 summarizes the responses given by the participants. This measure is referred to as self-assessment.

Table 21 - Number of participants indicating a level of programming experience (self-assessment)

Self-Assessment	0	0.5	1	1.5	2	3
Number of Participants	18	3	15	4	5	2

Finally, the participants were asked if they had experience using Logo or the related Lego Mindstorms. Two participants had used Logo and eight had used Lego Mindstorms. One participant had used both Logo and Lego Mindstorms.

As can be seen, there were a range of experiences in terms of courses, previous use of Logo or Mindstorms use, and the participants' self-assessments. It was hypothesised that the participants' self-assessment of their experience would be influenced by their course work experience, their use of Logo or Mindstorms, and other programming experience. It will be shown that there is evidence that the self-assessment includes information from these other measures. Figure 20 shows, via four sunflower plots, the relation between each participant's self-assessment and the four other measures.

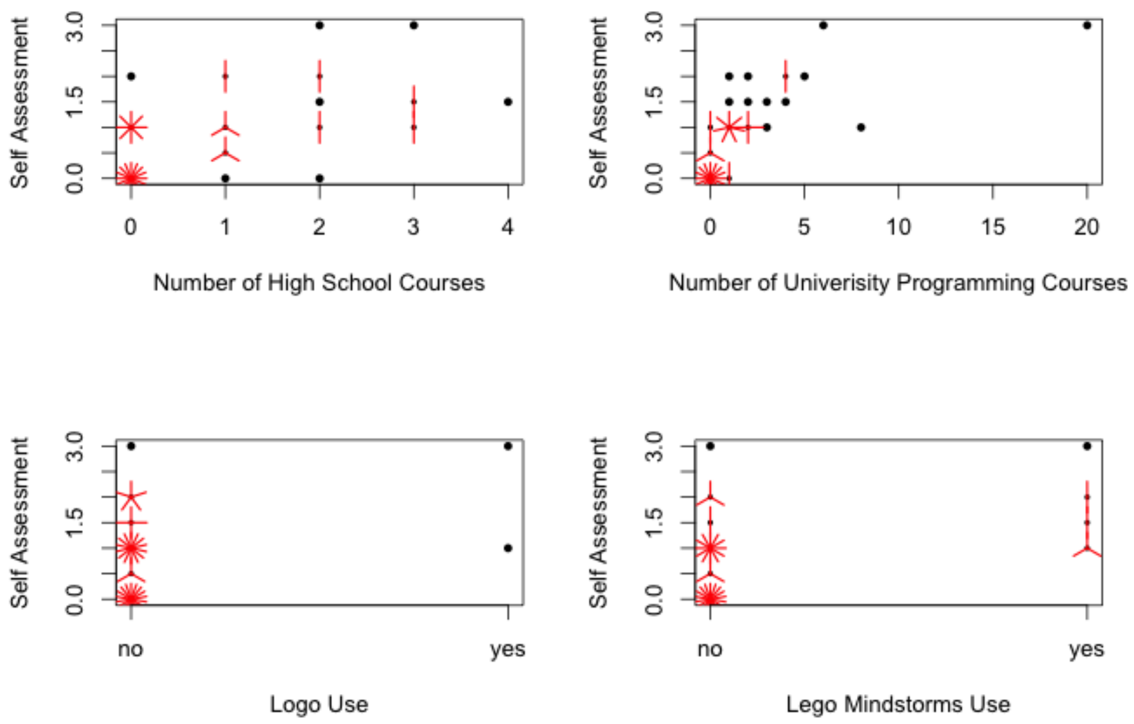


Figure 20 - Sunflower plots for various assessments of experience vs self-assessment

The correlations between self-assessment and each of the other measures are given in Table 22. The results in the table show that there is a significant positive correlation between each measure and self-assessment.

Table 22 - Pearson Correlations between Self-Assessment and each other measure

Measure	Pearson Correlation	p value
Number of High School Courses	0.578	<0.001
Number of University Courses	0.691	<0.001
Logo Use	0.305	0.037
Lego Mindstorms Use	0.448	0.002

The above shows that self-assessment may incorporate some of the information from the other assessments. As a test of this hypothesis the multiple regression model in Equation 1 can be tested. For this model, a 0,1 coding is used to indicate no or some experience with both Logo and Lego Mindstorms.

Equation 1

$$\begin{aligned}\mathbf{Self\ Assessment_i} &= \mathbf{b_0 + b_{High\ School}\ High\ School\ Courses_i + b_{University} University\ Courses_i} \\ &\quad \mathbf{+ b_{Logo} Logo_i + b_{Mindstorms} Mindstorms_i}\end{aligned}$$

Fitting the above model to the participants' data gives the following parameter estimates. The adjusted R^2 value is 0.634 with a p value of < 0.001.

Table 23 – Parameter estimates for Equation 1

Parameter	Estimate	Std. Error	t value	p value
b_0	0.317	0.095	3.340	0.002*
$b_{\text{High School}}$	0.163	0.076	2.140	0.038*
$b_{\text{University}}$	0.138	0.025	5.590	< 0.001*
b_{Logo}	0.419	0.379	1.110	0.275
$b_{\text{MindStorms}}$	0.608	0.207	2.940	0.005*

In the above model, Logo use does not contribute significantly to the model beyond the other measures, so Equation 2, with Logo removed, might be a better model.

Equation 2

$$\begin{aligned}\mathbf{Self\ Assessment_i} &= \mathbf{b_0 + b_{High\ School}\ High\ School\ Courses_i + b_{University} University\ Courses_i} \\ &\quad \mathbf{+ b_{Mindstorms} Mindstorms_i}\end{aligned}$$

Fitting the Equation 2 to the participants' data gives the following parameter estimates. The adjusted R^2 value is 0.632 with a p value of < 0.001.

Table 24 – Parameter estimates for Equation 2

Parameter	Estimate	Std. Error	t value	p value
b_0	0.313	0.095	3.290	0.002*
$b_{\text{High School}}$	0.182	0.074	2.450	0.019*
$b_{\text{University}}$	0.139	0.025	5.610	<0.001*
$b_{\text{MindStorms}}$	0.629	0.206	3.050	0.004*

The above analysis shows that there is a relationship between self-assessment and the other measures of experience and indicated that the variation in course work and Lego Mindstorms experience accounts for about 63% of the variation in self-assessment. Therefore, the hypothesis that the self-assessment includes the participants' consideration of their experience with programming courses and Lego Mindstorms, along with other experience, has supporting evidence.

In Clarke et al. (2005), a four-point self-assessment scale was used to assess experience with the software being used in the study. In that study, the authors divided the participants into two groups of high and low experience, with the bottom two positions on the scale being in the low experience group. The results of that study showed that such a division revealed a difference in effect of instructional materials depending on experience. Therefore, I similarly divided the participants into low and high programming experience, with the No and Some Experience (those with self-assessment ≤ 1) participants in a low experience group and the remaining participants (those with self-assessment > 1) in the high experience group. Table 27 shows the number of participants in each self-assessment level and condition.

Table 25 - Number of participants by condition and self-assessment level

Condition / Self-assessment	High Cognitive Load	Low Cognitive Load
all	24	23
≤ 1	18	18
> 1	6	5

In the next sections, the timings for the study sections and individual problems are presented, and the different experience levels will help explain those results.

6.3.2 Timings and Success

This section presents the number of problems the participants completed successfully in each study section and times the participants spent completing each section. Then the times the participants spent completing each individual problem are examined.

CLT's guidance fading and expertise reversal effects predict that some cognitive supports may be less effective, or even detrimental, to the learning of students who have higher levels of experience. Therefore, including higher experience participants in a comparison may hide a difference of effect between instructional conditions for the lower experience participants. Therefore, in each presentation of success and timings, three separate presentations are made, one for all participants, one for participants with a self-assessment ≤ 1, and one for participants with a self-assessment > 1, to see if there is an effect of the different cognitive load conditions hidden by the higher experience participants.

The measurement of completion times poses some difficulty and warrants some discussion before presenting the results. Since participants knew if their solution matched the problem goal, it can be assumed that a participant did not complete a problem thinking he or she had the right answer when he had an incorrect solution, so when interpreting the results the ideas of "did not succeed at a problem", "failed at a problem", and "gave up on a problem" are synonymous.

Including the times of participants who did not complete a problem poses some difficulty in interpreting results, because it is unknown if the additional time that the participants who gave up on a problem would have needed to complete the problem was independent of the condition they were assigned to. Because there is not a way to reliably estimate the length of time the participants who gave up would have taken to complete a problem, the times for participants who gave up are excluded when analysing the timings for the individual problems.

The problem of timing becomes more complex when considering timing of sections, when multiple problems are considered. Using the above method when considering the section timings would involve removing the timings for some participants, meaning the section timings would then be based on different problems for different participants. On the other hand, excluding any participant who did not complete at least one problem would result in many participants being excluded from some section timings (Number excluded in this case: Subprogram Instruction: 8/47, Subprogram Test: 18/47, Loop Instructions: 14/47, Loop Test: 21/47). Furthermore, including all participants gives a sense of how long participants spent doing sections of the study, regardless of their success or failure at individual problems. Therefore, timings at the section level are obtained by including timings for all problems, regardless of if the participants completed individual problems or not.

The raw timings for each problem or section, for either the instructional material or for the tests, are taken from the time from when the participant took his or her first action in creating a Logo program until he or she took a final action. At that point, the participant either succeeded at the problem or gave up.

There are 10 instances where participants did not attempt a problem. One participant did not attempt the last 5 problems, as she did not feel she could complete them. In one case the software failed part way through a problem, so the participant did not complete the problem; however, the attempt cannot fairly be called a failure. The reasons for the other 4 non-attempts are not known. The non-attempts are discussed later in the results for the individual problems.

6.3.2.1 Section Success and Timings

In this section, the number of successes and timings that the participants had over each section of the study are presented. The results are presented for all participants and for the participants in each of the two self-assessment levels, ≤ 1 and > 1. The timings for each participant are shown in Appendix O and the results for the successes for each section are found in Appendix P through Appendix S. For each comparison of successes and times, a Mann-Whitney U test is used, because, in some cases, the distribution of success and time are not normal.

6.3.2.1.1 All Participants

Table 26 shows the distribution of number of successes and the median number of successes for each section of the study by condition along with results of a Mann-Whitney U test on each section result to test for a significant difference between conditions based on the number of successes for each participant. Because the introduction was not divided into separate problems, that section is not included in the success results.

Table 26 - Analysis of number of successes for each section of the study.

The distribution of number of problem successes (high cognitive load on the top row, low cognitive load on the bottom), the medians of number of successes for each condition, and the results of a Mann-Whitney U test comparing the conditions. For each test, $\mathbf{n_{High} = 24, n_{Low} = 23}$.

Section	Problem Count	Condition	Distribution of Successes						Median Success Rate	U	p-value
			1	2	3	4	5	6			
Subprogram Instruction	6	High	0	0	0	0	3	21	6	306	0.337
		Low	0	0	0	3	2	18	6		
Subprogram Test	5	High	0	1	2	7	14		5	241	0.386
		Low	1	0	0	6	16		5		
Loop Instruction	5	High	0	0	5	3	16		5	247	0.459
		Low	0	1	0	5	17		5		
Loop Test	6	High	3	1	2	1	2	15	6	296	0.648
		Low	1	1	2	3	5	11	5		

Table 27 shows the average timings for the instructional and testing sections of the study by condition.

Table 27 - Median timings by condition for each section of Logo study.

For each test, $n_{High} = 24, n_{Low} = 23$.

Section	High Cognitive Load Time	Low Cognitive Load Time	Difference	U	p-value
Introduction	720.5s	696s	24.5s	326.5	0.287
Subprograms Instruction	932s	824s	108s	345	0.146
Subprograms Test	1744s	1671s	73s	275	0.992
Loop Instruction	1370.5s	834s	536.5s	447	<0.001
Loop Test	2411.5s	1992s	419.5s	331.5	0.242

The only significant difference in timings between conditions was for the loop instruction section (U = 447, p < 0.001). The other three sections did not show a significant difference in time (Introduction: U = 326.5, p = 0.287; Subprograms Instruction: U = 345, p = 0.146; Subprograms Test: U = 275, p = 0.992; Loop Test: U = 331.5, p = 0.242, for all tests $n_{High} = 24, n_{Low} = 23$).

6.3.2.1.2 Participants with Self-Assessment ≤ 1

As discussed, experience levels may play a role in the effect of cognitive load interventions, so the next two subsections present the results for the two self-assessment levels, starting with participants with a low self-assessment level ≤ 1.

Table 28 shows the distribution of number of successes and the median number of successes for each section of the study by condition along with results of a Mann-Whitney U test on each section result to test for a significant difference between conditions based on the number of successes for each participant.

Table 28 - Analysis of number of successes for participants with self-assessment ≤ 1 for each section of the study.

The distribution of number of problem successes (high cognitive load on the top row, low cognitive load on the bottom), the medians of number of successes for each condition, and the results of a Mann-Whitney U test comparing the conditions. For each test, $\mathbf{n_{High} = 18, n_{Low} = 18}$.

Section	Problem Count	Condition	Distribution of Successes						Median Success Rate	U	p-value
			1	2	3	4	5	6			
Subprogram Instruction	6	High	0	0	0	0	3	15	6	175	0.551
		Low	0	0	0	3	1	14	6		
Subprogram Test	5	High	0	1	2	7	8		4	123	0.177
		Low	1	0	0	5	12		5		
Loop Instruction	5	High	0	0	5	3	10		5	135	0.345
		Low	0	1	0	5	12		5		
Loop Test	6	High	3	1	2	1	2	9	5.5	157	0.893
		Low	1	1	2	3	3	8	5		

Table 29 shows the average timings for the instructional and testing sections of the study by condition for participants with self-assessment ≤ 1.

Table 29 - Median timings for participants with self-assessment ≤ 1 by condition for each section of Logo study.

For each test, $n_1 = 18, n_2 = 18$.

Section	High Cognitive Load Time	Low Cognitive Load Time	Difference	U	p-value
Introduction	747s	736s	11s	181	0.558
Subprograms Instruction	968s	825s	143s	240	0.013
Subprograms Test	1995.5s	1699.5s	296s	171	0.791
Loop Instruction	1467.5s	846s	621.5s	274	<0.001
Loop Test	2460s	2026s	434s	201.5	0.217

For the participants with self-assessment ≤ 1, both the subprogram instructions and Loop instructions resulted in a significant difference in time taken between conditions (Subprogram Instruction: U = 240, $p < 0.013$; Loop Instruction: U = 247, $p < 0.001$, for all tests $n_1 = 18, n_2 = 18$).

6.3.2.1.3 Participants with Self-Assessment > 1

In this subsection, the same results are presented, but for participants with Self-Assessment > 1. Table 30 shows the distribution of number of successes and the median number of successes for each section of the study by condition along with results of a Mann-Whitney U test on each section result to test for a significant difference between conditions based on the number of successes for each participant.

Table 30 - Analysis of number of successes for each section of the study for participants with self-assessment > 1.

The distribution of number of problem success (high cognitive load on the top row, low cognitive load on the bottom), the medians of number of successes for each condition, and the results of a Mann-Whitney U test comparing the conditions. For each test, $n_{High} = 6, n_{Low} = 5$.

Section	Problem Count	Condition	Distribution of Successes (High/Low Condition)						Median Success Rate	U	p-value
			1	2	3	4	5	6			
Subprogram Instruction	6	High	0	0	0	0	0	6	6	18	0.361
		Low	0	0	0	0	1	4	6		
Subprogram Test	5	High	0	0	0	0	6		5	18	0.363
		Low	0	0	0	1	4		5		
Loop Instruction	5	High	0	0	0	0	6		5	15	1
		Low	0	0	0	0	5		5		
Loop Test	6	High	0	0	0	0	0	6	6	21	0.134
		Low	0	0	0	0	2	3	6		

Table 31 shows the average timings for the instructional and testing sections of the study by condition for participants with self-assessment > 1.

Table 31 - Median timings for participants with self-assessment > 1 by condition for each section of Logo study.

For each test, $n_{High} = 6, n_{Low} = 5$.

Section	High Cognitive Load Time	Low Cognitive Load Time	Difference	U	p-value
Introduction	674.5s	592s	82.5s	21	0.329
Subprograms Instruction	656s	715s	-59s	8	0.247
Subprograms Test	1204s	1296s	-92s	13	0.792
Loop Instruction	885.5s	758s	127.5s	23	0.177
Loop Test	1866s	1326s	540s	18	0.662

As can be seen in Table 31, for participants with self-assessment > 1, none of the sections of the study resulted in a significant time difference between conditions (Introduction: U = 21, p = 0.329; Subprogram Instruction: U = 8, p = 0.247; Subprograms Test: U = 13, p = 0.729; Loop Instruction: U = 23, p = 0.177; Loop Test: U = 18, p = 0.662, for all tests $n_{High} = 6, n_{Low} = 5$).

6.3.2.1.4 Summary Section Results

In this section, the significant results from the analysis of the study sections are reviewed.

No significant difference in success rates at the problems in any of the sections detected; however, some significant differences were detected in the timings between conditions for some sections of the study.

The differences in timings between conditions were significant the subprograms and loop instructions for participants with self-assessment ≤ 1. For that experience level, the participants in the high cognitive load condition completed the subprograms instruction with a median time of 968s (16m:8s) while those in the low cognitive load condition completed with a median time of 825s (13m 45s). This difference of 2m:23s (significant at p = 0.013) is about a 15% savings in time. When considering participants with a self-assessment level > 1, the high cognitive load group had a median time of 656s (10m:56s) and the low cognitive load group had a median time of 715s (11m:55s). The difference between the high and low cognitive load groups' median time was 59s, with the high cognitive load group having the smaller median time; however, this difference was not significant ($p = 0.792$).

In the loop instruction section of the study the study, the high cognitive load condition participants took a median time of 1370.5s (22m:50s) while the participants in the low cognitive load condition took a median time of 834s (13m:64s). Therefore, the participants in the low cognitive load condition saved about 536.5s (10m:56s) (significant at $p < 0.001$) or 39% over the high cognitive load condition participants. The participants of with self-assessment ≤ 1 in the high cognitive load condition took a median time of 1467.5s (24m:27s) while the participants in the low cognitive load condition took a median time of 846s (14m:6s). Therefore, the participants in the low cognitive load condition saved about 621.5s (10m:21s) (significant at $p < 0.001$) or 42% over the high cognitive load condition participants, a pretty large savings. For participants with self-assessment > 1, the high cognitive load group had a median time of 885.5s (14m:45s) and the low cognitive load group has a median time of 758s (12m:38s). For those participant, the difference between the high and low cognitive load groups' median time was 127.5s (2m:07s); however, this difference was not significant ($p = 0.177$).

From the lack of significant difference in performance in the test sections, both in terms of time and success rate, between participants in the two conditions, it appears, at this high level of analysis, the participants with self-assessment ≤ 1 in the low cognitive load condition, despite taking less time to complete the instructional sections, performed as well as those in the high cognitive load condition in terms of success rate (Subprograms Test: $p = 0.177$; Loop Test: $p = 0.893$) and time (Subprograms Test: $p = 0.791$; Loop Test: $p = 0.217$).

These results point to the usefulness of some of the cognitive supports provided to the participants in the low cognitive load condition during instruction, particularly when the participants were sufficiently inexperienced, as the participants with self-assessment ≤ 1 in the low-cognitive load condition performed as well on the tests as those in the high cognitive load condition, but took less time in training. Likewise for the loop instructions, with participants of all experience levels considered, the participants in the low cognitive load condition took less time on the instructions (22m:50s vs 13m:64s, $p < 0.001$); however, there was no significant difference in test success rates (Subprograms Test: $p = 0.386$; Loop Test: $p = 0.648$) or times (Subprograms Test: $p = 0.992$; Loop Test: $p = 0.242$). It would be useful to determine if the participants learned the same things in both instructional conditions, allowing them to perform equally well on the tests, despite, in some cases, taking less time during instruction. To further examine the effects of the different instructional conditions on participant learning, the next sections present success and time results for each individual problem and then examine in more detail for some particular interesting problems and pairs of problems how the participants completed the problems and possible effects of instructional type on those strategies.

6.3.2.2 Results for Individual Problems

In this section, results are given for the individual problems in the four sections of the study where the different instructional conditions potentially had an effect: subprogram instruction, subprogram test, loop instruction, and loop test. For each problem, it is shown, by instructional condition and experience, how many participants successfully completed each problem and how long the problems took for participants who were successful at the problem. It is also discussed how many participants did not attempt the problems and possible reasons for the non-attempts. The successes and failures are tested for dependence on instructional condition using Fisher exact tests; non-attempts are excluded from this analysis. The timings for each problem are also presented, again divided by experience level. The times include only participants successful at the respective problems, and an effect of condition on time is tested for using a Mann-Whitney U test.

For each comparison of timings between conditions, a Mann-Whitney U test is used, because the times are sometimes non-normal in their distribution. Only participants who successfully completed a problem are included in this analysis.

6.3.2.2.1 Sub Program Instruction

There are five problems in the sub programs instruction: Square, Pentagon, Radiation, Bow Tie, and house. The data for all results in the sub program instruction can be found in Appendix P.

6.3.2.2.1.1 Square

The number of successes and failures for the Square problem, by experience level and condition, are shown in Table 32. There was no dependence detected between success and condition for any level of experience (all: $p = 1$; ≤ 1: $p = 1$; > 1: $p = 1$).

Table 32 – Successes, failures, and non-attempts for the Square problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 33 shows the median times for the Square problem by experience level and condition for participants who were successful at the problem. Table 33 shows that there was a significant difference in times between the conditions for participants with a self-assessment ≤ 1 (U = 245, $n_{\mathrm{High}} = 18, n_{\mathrm{Low}} = 18$, p = 0.033), with participants in the low cognitive load condition having a smaller median completion time. The difference between the conditions was not significant, for self-assessment > 1 (U = 11.5, $n_{\mathrm{High}} = 6, n_{\mathrm{Low}} = 5$, p=0.582).

Table 33 – Median times for the Square problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	94s	78s	16s	345.5	0.142
≤ 1	99s	81.5s	17.5s	245	0.033
> 1	63s	75s	-12s	11.5	0.582

6.3.2.2.1.2 Pentagon

The number of successes and failures for the Pentagon problem, by experience level and condition, are shown in Table 34. There was no dependence detected between success and condition for any level of experience (all: p = 1; $\leq$ 1: p = 1; $>$ 1: p = 1).

Table 34 – Successes, failures, and non-attempts for the Pentagon problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 35 shows the median times for the Pentagon problem by experience level and condition for participants who were successful at the problem. There was a significant difference in times between the conditions for all participants (U = 399, $n_{High} = 24, n_{Low} = 23$, p = 0.009) and participants with a self-assessment ≤ 1 (U = 253, $n_{High} = 18, n_{Low} = 18$, p = 0.004), with participants in the low cognitive load condition having a smaller median completion time. The difference between the conditions was not significant, for self-assessment > 1 (U = 15, $n_{High} = 6, n_{Low} = 5$, p = 1).

Table 35 – Median times for the Pentagon problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	96.5s	77s	19.5s	399	0.009
≤ 1	96.5s	75s	21.5s	253	0.004
> 1	96.5s	83s	13.5s	15	1

6.3.2.2.1.3 Radiation

The number of successes and failures for the Radiation problem, by experience level and condition, are shown in Table 36. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 36 – Successes, failures, and non-attempts for the Radiation problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 37 shows the median times for the Radiation problem by experience level and condition for participants who were successful at the problem. There was a significant difference in times between the conditions for all participants (U = 374, $n_{\text{High}} = 24, n_{\text{Low}} = 23$, p = 0.038) and participants with a self-assessment ≤ 1 (U = 230.5, $n_{\text{High}} = 18, n_{\text{Low}} = 18$, p = 0.031), with participants in the low cognitive load condition having a smaller median completion time. The difference between the conditions was not significant for self-assessment > 1 (U = 19, $n_{High} = 6, n_{Low} = 5$, p = 0.552).

Table 37 – Median times for the Radiation problem by condition and personal experience for participants who were successful at the problem.

The results of a Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	48.5s	40s	8.5s	374	0.038
≤ 1	59s	42s	17s	230.5	0.031
> 1	42s	37s	5s	19	0.522

6.3.2.2.1.4 H

The number of successes and failures for the H problem, by experience level and condition, are shown in Table 38. For all participants, there were five participants in the low cognitive load condition who did not complete the problems successfully, compared to no participants in the high cognitive load condition. A Fisher exact test revealed a significant dependence between condition and success rate (p = 0.022). For the different experience levels there were more participants in the low cognitive load condition who did not complete the problem successfully; however, for the separate experience levels the Fisher exact tests did not detect a significant dependence on condition (≤ 1: $p = 0.104$; > 1: p = 0.455).

Table 38 – Successes, failures, and non-attempts for the H problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	0.022
	Low	0	18	5	
≤ 1	High	0	18	0	0.104
	Low	0	14	4	
> 1	High	0	6	0	0.455
	Low	0	4	1	

Table 39 shows the median times for the H problem by experience level and condition for participants who were successful at the problem. For this problem, the participants who were successful at both experience levels and all participants had a significantly shorter median completion time in the high cognitive load condition (all: U = 62.5, $n_{High} = 24, n_{Low} = 18$, p < 0.001; ≤ 1: U = 45, $n_{High} = 18, n_{Low} = 14$, p = 0.002; > 1: U = 1, $n_{High} = 6, n_{Low} = 4$, p = 0.023).

Table 39 – Median times for the H problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	130s	237s	-107s	62.5	<0.001
≤ 1	133.5s	221s	-87.5s	45	0.002
> 1	109.5s	248.5s	-139s	1	0.023

6.3.2.2.1.5 Bow Tie

The number of successes and failures for the Bow Tie problem, by experience level and condition, are shown in Table 40. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 40 – Successes, failures, and non-attempts for the Bow Tie problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	21	3	1
	Low	0	20	3	
≤ 1	High	0	15	3	1
	Low	0	15	3	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 41 shows the median times for the Bow Tie problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 256.5, $n_{\text{High}} = 24, n_{\text{Low}} = 23$, p = 0.23; ≤ 1: U = 143.5, $n_{\text{High}} = 18, n_{\text{Low}} = 18$, p = 0.206; U = 17, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.792).

Table 41 – Median times for the Bow Tie problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	191s	132s	59s	256.5	0.23
≤ 1	240s	135s	105s	143.5	0.206
> 1	140s	123s	17s	17	0.792

6.3.2.2.1.6 House

The number of successes and failures for the House problem, by experience level and condition, are shown in Table 42. There was no dependence detected between success and condition for any level of experience (all: p = 1; $\leq$ 1: p = 1; $>$ 1: p = 1).

Table 42 – Successes, failures, and non-attempts for the House problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 43 shows the median times for the House problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 297.5, $n_{High} = 24, n_{Low} = 23$, p = 0.655; $\leq$ 1: U = 178.5, $n_{High} = 18, n_{Low} = 18$, p = 0.613; U = 17, $n_{High} = 6, n_{Low} = 5$, p = 0.792).

Table 43 – Median times for the House problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	90.5s	74s	16.5s	297.5	0.655
≤ 1	123.5s	92.5s	31s	178.5	0.613
> 1	71s	43s	28s	17	0.792

6.3.2.2.2 Subprograms Test

The next section of the study, the subprograms test, required the participants to complete five problems. The same test was given to all participants, regardless of their instructional condition. The participants were asked not to look at how they completed their previous programs or look at the instruction material again. However, the participants were told they could reuse existing programs, except for Bow Tie, as reusing that program made the first test program in the test too easy. There were five problems in the subprograms test: Bow Tie Two, K, Up Arrow, Star, and Blade. The test can be found in Appendix L, and Appendix Q shows the results for all the problems in the subprograms test.

6.3.2.2.2.1 Bow Tie Two

The Bow Tie Two program required participants to complete the same figure as in Bow Tie. Before starting the subprograms test, the participants were told they could not use Bow Tie to complete Bow Tie Two, as doing so made Bow Tie Two too easy, though they could use any other program. Two participants used Bow Tie anyway, making the problem very easy (these participants completed the problem in 9s and 41s, the two fastest times). Since, the participants did not complete the problem as intended, their results are not included in the analysis and are counted as non-attempts. There were two other participants who did not attempt the problem. It is possible that these participants were confused by the instructions not to reuse Bow Tie and thought they did not need to complete the problem at all. The number of successes and failures for the Bow Tie Two problem, by experience level and condition, are shown in Table 44. There was no dependence detected between success and condition for any level of experience (all: p = 0.669; $\leq$ 1: p = 0.672; $>$ 1: p = 1).

Table 44 – Successes, failures, and non-attempts for the Bow Tie Two problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	4	18	2	0.669
	Low	0	19	4	
$\leq$ 1	High	4	12	2	0.672
	Low	0	14	4	
$>$ 1	High	0	6	0	1
	Low	0	5	0	

Table 41 shows the median times for the Bow Tie Two problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 182, $n_{\text{High}} = 18, n_{\text{Low}} = 19$, p = 0.75; $\leq$ 1: U = 85.5, $n_{\text{High}} = 12, n_{\text{Low}} = 14$, p = 0.959; U = 18.5, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.583).

Table 45 – Median times for the Bow Tie Two problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	149	138	11	182.0	0.75
$\leq$ 1	172	140.5	31.5	85.5	0.959
$>$ 1	137	85	52	18.5	0.583

6.3.2.2.2.2 K

The number of successes and failures for the K problem, by experience level and condition, are shown in Table 46. There was no dependence detected between success and condition for any level of experience (all: p = 0.665; ≤ 1: p = 0.66; > 1: p = 1).

Table 46 – Successes, failures, and non-attempts for the K problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	22	2	1
	Low	0	22	1	
≤ 1	High	0	16	2	1
	Low	0	17	1	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 47 shows the median times for the K problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 283.5, $n_{\text{High}} = 22, n_{\text{Low}} = 22$, p = 0.336; ≤ 1: U = 160.5, $n_{\text{High}} = 16, n_{\text{Low}} = 17$, p = 0.387; U = 17, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.784).

Table 47 – Median times for the K problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
All	192.5s	160.5s	32s	283.5	0.336
≤ 1	219.5s	170s	49.5s	160.5	0.387
> 1	159.5s	116s	43.5s	17	0.784

6.3.2.2.2.3 Up Arrow

The number of successes and failures for the Up Arrow problem, by experience level and condition, are shown in Table 48. There was no dependence detected between success and condition for any level of experience (all: p = 0.609; ≤ 1: p = 1; > 1: p = 1).

Table 48 – Successes, failures, and non-attempts for the Up Arrow problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	23	1	0.609
	Low	0	21	2	
≤ 1	High	0	17	1	1
	Low	0	16	2	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 49 shows the median times for the Up Arrow problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 209.5, $n_{\mathrm{High}} = 23, n_{\mathrm{Low}} = 21$, p = 0.495; ≤ 1: U = 106, $n_{\mathrm{High}} = 17, n_{\mathrm{Low}} = 16$, p = 0.292; U = 15, $n_{\mathrm{High}} = 6, n_{\mathrm{Low}} = 5$, p = 1).

Table 49 – Median times for the Up Arrow problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	242s	248s	-6s	209.500	0.459
≤ 1	242s	280s	-38s	106.000	0.292
> 1	199.5s	204s	-4.5s	15.000	1

6.3.2.2.2.4 Star

The number of successes and failures for the Star problem, by experience level and condition, are shown in Table 50. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 50 – Successes, failures, and non-attempts for the Star problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	23	1	1
	Low	0	22	1	
≤ 1	High	0	17	1	1
	Low	0	17	1	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 51 shows the median times for the Star problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 260, $n_{\text{High}} = 23, n_{\text{Low}} = 22$, p = 0.883; ≤ 1: U = 161, $n_{\text{High}} = 17, n_{\text{Low}} = 17$, p = 0.586; U = 15, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 1).

Table 51 – Median times for the Star problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	304s	276s	28s	260	0.883
≤ 1	332s	327s	5s	161	0.586
> 1	197s	197s	0s	15	1

6.3.2.2.2.5 Blade

The Blade problem was divided into two steps. In the first step, the participants were asked to draw the top half of the image in one program and then the whole image in another program. The results presented in this section examine if each participant was able to complete the whole image and how long it took them to complete the whole image, including the time to draw the top half, even if he or she did not reuse the top half when completing the final image. The times are based on the total times the participants used on both halves of the problem. There were two participants who did not attempt the second half of the problem, but they did attempt the first half of the problem, so their results are still included as an attempt at the Blade problem. In section 6.3.3.1.3, the participants' reuse of the half image as a subprogram in completing the final problem is examined, and there the relations between success at the first half of the problem to success in the second half is explored; therefore, in that section the two participants who did not attempt the second half of the problem are discussed.

The number of successes and failures for the Blade problem, by experience level and condition, are shown in Table 52. There was no dependence detected between success and condition for any level of experience (all: p = 0.461; ≤ 1: p = 0.228; > 1: p = 0.455).

Table 52 – Successes, failures, and non-attempts for the Blade problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	18	6	0.461
	Low	0	20	3	
≤ 1	High	0	12	6	0.228
	Low	0	16	2	
> 1	High	0	6	0	0.455
	Low	0	4	1	

Table 53 shows the median times for the Blade problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 133, $n_{High} = 18, n_{Low} = 20$, p = 0.176; ≤ 1: U = 86, $n_{High} = 12, n_{Low} = 16$, p = 0.664; U =5, $n_{High} = 6, n_{Low} = 4$, p = 0.171).

Table 53 – Median times for the Blade problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	491.5s	614.5s	-123s	133	0.176
≤ 1	600s	614.5s	-14.5s	86	0.664
> 1	366s	680s	-314s	5	0.171

6.3.2.2.3 Loop Instructions

After learning about and being tested on subprograms, the participants were introduced to the Repeat command, which is a basic loop command. There are five problems in the loop instructions: New Square, Radioactive 2, Wall, Circle, and Starburst. The data for all results in the loop instructions can be found in Appendix R.

6.3.2.2.3.1 New Square

The number of successes and failures for the New Square problem, by experience level and condition, are shown in Table 54. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 54 – Successes, failures, and non-attempts for the New Square problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 55 shows the median times for the New Square problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 247, $n_{High} = 24, n_{Low} = 23$, p = 0.544; ≤ 1: U = 158.5, $n_{High} = 18, n_{Low} = 18$, p = 0.924; U = 9, $n_{High} = 6, n_{Low} = 5$, p = 0.329).

Table 55 – Median times for the New Square problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	69.5s	71s	-1.5s	247	0.544
≤ 1	76s	82.5s	-6.5s	158.5	0.924
> 1	60s	68s	-8s	9	0.329

6.3.2.2.3.2 Radioactive 2

The number of successes and failures for the Radioactive 2 problem, by experience level and condition, are shown in Table 56. One participant did not attempt this problem. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 56 – Successes, failures, and non-attempts for the Radioactive 2 problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
All	High	0	24	0	1
	Low	1	22	0	
≤ 1	High	0	18	0	1
	Low	1	17	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 57 shows the median times for the Radioactive 2 problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 281, $n_{\text{High}} = 24, n_{\text{Low}} = 22$, p = 0.717; ≤ 1: U = 149, $n_{\text{High}} = 18, n_{\text{Low}} = 17$, p = 0.908; U = 18, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.645).

Table 57 – Median times for the Radioactive 2 problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	40.5s	38.5s	2s	281	0.717
≤ 1	40.5s	40s	0.5s	149	0.908
> 1	40s	30s	10s	18	0.645

6.3.2.2.3.3 Wall

The number of successes and failures for the Wall problem, by experience level and condition, are shown in Table 58. There was no dependence detected between success and condition for any level of experience (all: p = 0.416; ≤ 1: p = 0.402; > 1: p = 1).

Table 58 – Successes, failures, and non-attempts for the Wall problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	19	5	0.416
	Low	0	21	2	
≤ 1	High	0	13	5	0.402
	Low	0	16	2	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 59 shows the median times for the Wall problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 253, $n_{High} = 19, n_{Low} = 21$, p = 0.151; ≤ 1: U = 143, $n_{High} = 13, n_{Low} = 16$, p = 0.091; U = 18, $n_{High} = 6, n_{Low} = 5$, p = 0.931).

Table 59 – Median times for the Wall problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	374s	308s	66s	253	0.151
≤ 1	395s	322.5s	72.5s	143	0.091
> 1	278s	301s	-23s	16	0.931

6.3.2.2.3.4 Circle

The number of successes and failures for the Circle problem, by experience level and condition, are shown in Table 60. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 60 – Successes, failures, and non-attempts for the Circle problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1.000
	Low	0	23	0	
≤ 1	High	0	18	0	1.000
	Low	0	18	0	
> 1	High	0	6	0	1.000
	Low	0	5	0	

Table 61 shows the median times for the Circle problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 261.5, $n_{High} = 24, n_{Low} = 23$, p = 0.766; $\leq$ 1: U = 174.5, $n_{High} = 18, n_{Low} = 18$, p = 0.704; U = 6, $n_{High} = 6, n_{Low} = 5$, p = 0.126).

Table 61 – Median times for the Circle problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	42s	45s	-3s	261.5	0.766
≤ 1	43s	46s	-3s	174.5	0.704
> 1	33.5s	45s	-11.5s	6	0.126

6.3.2.2.3.5 Starburst

The number of successes and failures for the Starburst problem, by experience level and condition, are shown in Table 62. There was no dependence detected between success and condition for any level of experience (all: p = 0.517; ≤ 1: p = 0.489; > 1: p = 1).

Table 62 – Successes, failures, and non-attempts for the Starburst problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	16	8	0.517
	Low	0	18	5	
≤ 1	High	0	10	8	0.489
	Low	0	13	5	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 63 shows the median times for the Starburst problem by experience level and condition for participants who were successful at the problem. There was a significant difference in times between instructional conditions for all experience levels (all: U = 258, $n_{\text{High}} = 16, n_{\text{Low}} = 18$, p < 0.001; ≤ 1: U = 119, $n_{\text{High}} = 10, n_{\text{Low}} = 13$, p < 0.001; U = 224.5, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.017).

Table 63 – Median times for the Starburst problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	578s	224.5s	353.5s	258	<0.001
≤ 1	889.5s	229s	660.5s	119	<0.001
> 1	393.5s	169s	224.5s	28	0.017

6.3.2.2.4 Loop Test

The next section of the study, the loop test, required the participants to complete five problems. As with the subprograms test, the same test was given to all participants, regardless of their instructional condition. Again, the participants were asked not to look at how they completed their previous programs or look at the instructional material. However, the participants were told they could reuse existing programs. There were six problems in the subprograms test: Hexagon, Humps, Saw Blade, Gear, Wave, and Tree. The test can be found in Appendix M.

Appendix S shows the results for all the problems in the sub program test.

6.3.2.2.4.1 Hexagon

The number of successes and failures for the Hexagon problem, by experience level and condition, are shown in Table 64. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; > 1: p = 1).

Table 64 – Successes, failures, and non-attempts for the Hexagon problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	0	24	0	1
	Low	0	23	0	
≤ 1	High	0	18	0	1
	Low	0	18	0	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 65 shows the median times for the Hexagon problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 273, $n_{High} = 24, n_{Low} = 23$, p = 0.958; ≤ 1: U = 156, $n_{High} = 18, n_{Low} = 18$, p = 0.862; U = 18, $n_{High} = 6, n_{Low} = 5$, p = 0.647).

Table 65 – Median times for the Hexagon problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	58s	59s	-1s	273	0.958
≤ 1	59.5s	72s	-12.5s	156	0.862
> 1	44.5s	55s	-10.5s	18	0.647

6.3.2.2.4.2 Bumps

The number of successes and failures for the Bumps problem, by experience level and condition, are shown in Table 66. Two participants did not attempt this problem. One participant, who struggled through much of the study, decided she did not want to try any more problems once she got to the Bumps problem. The other participant's reason for not trying the problem is unknown. There was no dependence detected between success and condition for any level of experience (all: p = 1; $\leq$ 1: p = 0.603; $>$ 1: p = 1).

Table 66 – Successes, failures, and non-attempts for the Bumps problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	1	21	2	1
	Low	1	21	1	
≤ 1	High	1	15	2	0.603
	Low	0	17	1	
> 1	High	0	6	0	1
	Low	1	4	0	

Table 67 shows the median times for the Bumps problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 245, $n_{\mathrm{High}} = 21, n_{\mathrm{Low}} = 21$, p = 0.546; ≤ 1: U = 151.5, $n_{\mathrm{High}} = 15, n_{\mathrm{Low}} = 17$, p = 0.375; U = 12, $n_{\mathrm{High}} = 6, n_{\mathrm{Low}} = 4$, p = 1).

Table 67 – Median times for the Bumps problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	140s	116s	24s	245	0.546
≤ 1	155s	119s	36s	151.5	0.375
> 1	117s	110.5s	6.5s	12	1

6.3.2.2.4.3 Saw Blade

The number of successes and failures for the Saw Blade problem, by experience level and condition, are shown in Table 68. The participant mentioned in the Bumps problem also did not attempt this problem, as she had given up by this point. There was no dependence detected between success and condition for any level of experience (all: p = 1; ≤ 1: p = 1; $>$ 1: p = 1).

Table 68 – Successes, failures, and non-attempts for the Saw Blade problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	1	18	5	1
	Low	0	18	5	
≤ 1	High	1	12	5	1
	Low	0	13	5	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 69 shows the median times for the Saw Blade problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 206, $n_{\mathrm{High}} = 18, n_{\mathrm{Low}} = 18$, p = 0.171; ≤ 1: U = 88, $n_{\mathrm{High}} = 12, n_{\mathrm{Low}} = 13$, p = 0.611; U = 26, $n_{\mathrm{High}} = 6, n_{\mathrm{Low}} = 5$, p = 0.052).

Table 69 – Median times for the Saw Blade problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	264.5s	164.5s	100s	206	0.171
≤ 1	281.5s	219s	62.5s	88	0.611
> 1	232.5s	130s	102.5s	26	0.052

6.3.2.2.4.4 Gear

The number of successes and failures for the Gear problem, by experience level and condition, are shown in Table 70. The participant mentioned in the Bumps problem also did not attempt this problem, as she had given up by this point. There was no dependence detected between success and condition for any level of experience (all: p = 0.459; ≤ 1: p = 0.264; > 1: p = 1).

Table 70 – Successes, failures, and non-attempts for the Gear problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	1	17	6	0.459
	Low	0	20	3	
≤ 1	High	1	11	6	0.264
	Low	0	15	3	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 71 shows the median times for the Gear problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 194, $n_{\text{High}} = 17, n_{\text{Low}} = 20$, p = 0.474; ≤ 1: U = 94, $n_{\text{High}} = 11, n_{\text{Low}} = 15$, p = 0.568; U = 19, $n_{\text{High}} = 6, n_{\text{Low}} = 5$, p = 0.537).

Table 71 – Median times for the Gear problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	277s	239s	38s	194	0.474
≤ 1	292s	239s	53s	94	0.568
> 1	243.5s	194s	49.5s	19	0.537

6.3.2.2.4.5 Wave

The number of successes and failures for the Wave problem, by experience level and condition, are shown in Table 72. Three participants did not attempt this problem. One participant did not complete this problem due to a software crash; his attempt is counted as a non-attempt, since it is not possible to know if he would have completed the problem. The participant mentioned in the Bumps problem also did not attempt this problem, as she had given up by this point. A third participant decided not to complete Wave and Tree. There was no dependence detected between success and condition for any level of experience (all: p = 0.521; ≤ 1: p = 0.728; > 1: p = 0.455).

Table 72 – Successes, failures, and non-attempts for the Wave problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	1	17	6	0.521
	Low	2	13	8	
≤ 1	High	1	11	6	0.728
	Low	2	9	7	
> 1	High	0	6	0	0.455
	Low	0	4	1	

Table 73 shows the median times for the Wave problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 110.5, $n_{\text{High}} = 17, n_{\text{Low}} = 13$, p = 1; $\leq$ 1: U = 55, $n_{\text{High}} = 11, n_{\text{Low}} = 9$, p = 0.704; U = 8.5, $n_{\text{High}} = 6, n_{\text{Low}} = 4$, p = 0.521).

Table 73 – Median times for the Wave problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
All	399s	391s	8s	110.5	1
≤ 1	438s	416s	22s	55	0.704
> 1	277.5s	335s	-57.5s	8.5	0.521

6.3.2.2.4.6 Tree

The number of successes and failures for the Tree problem, by experience level and condition, are shown in Table 74. The participant mentioned in the Bumps problem also did not attempt this problem, as did the one who gave up at Wave. There was no dependence detected between success and condition for any level of experience (all: p = 1; $\leq$ 1: p = 1; $>$ 1: p = 1).

Table 74 – Successes, failures, and non-attempts for the Tree problem by experience level and condition.

The p values are the result of a Fisher exact test for dependence of successes and failures on condition for each self-assessment level.

Self-Assessment	Condition	Non Attempts	Successes	Failures	p value
all	High	1	18	5	1
	Low	1	17	5	
≤ 1	High	1	12	5	1
	Low	1	12	5	
> 1	High	0	6	0	1
	Low	0	5	0	

Table 75 shows the median times for Tree problem by experience level and condition for participants who were successful at the problem. There was no significant difference in times between instructional conditions for any experience level (all: U = 188, $n_{\mathrm{High}} = 18, n_{\mathrm{Low}} = 17$, p = 0.258; $\leq$ 1: U = 99, $n_{\mathrm{High}} = 12, n_{\mathrm{Low}} = 12$, p = 0.128; U = 16, $n_{\mathrm{High}} = 6, n_{\mathrm{Low}} = 5$, p = 0.931).

Table 75 – Median times for the Tree problem by condition and personal experience for participants who were successful at the problem.

The results of Mann-Whitney U tests used to detect a significant difference in mean times are shown.

Self-Assessment	Median High Cognitive Load Time	Median Low Cognitive Load Time	Difference	U	p value
all	725.5s	608s	117.5s	188	0.258
≤ 1	895.5s	699s	196.5s	99	0.128
> 1	526.5s	530s	-3.5s	16	0.931

6.3.2.2.5 Problem Timing and Success Summary

In this subsection, the statistically significant results from the successes and timings of the problems are summarized.

Table 76 summarizes for what sections and problems there was a significant difference of times between high and low condition for the different levels of experience. A green cell indicated that the median time for the low cognitive load group was shorter and red indicates the high cognitive load group had a smaller median time. A dark colour indicates a significant difference in times between the conditions.

Only a few problems showed a significant difference of instructional condition on problem completion time. These problems were Square, Pentagon, Radiation, H, and Starburst. In the Square problem, the participants with self-assessment ≤ 1 had shorter times in the low cognitive load condition ($p = 0.033$). In Pentagon, Radiation, and Starburst the low cognitive load group had shorter times for all levels of experience (Pentagon: $p = 0.009$; Radiation: $p = 0.038$; Starburst: $p < 0.001$). In completing the H problem, participants in the high cognitive load condition performed better, regardless of self-assessment level ($p < 0.001$).

The problem that demonstrated the largest significant difference was the Starburst problem, where the participants in the high cognitive load condition had a median time of 578s (9m:38s) and those in the low cognitive load condition had a median time of 224.5s (3m:44s). The difference of 353.5s (5m:53s) was significant ($p < 0.001$). Among those with a self-assessment ≤ 1, the difference was even more pronounced, with the high cognitive load group having a median time of 889.5s (14m:49s) and those in the low cognitive load group 229s (3m:49s), a difference of 660.5s (11m), ($p < 0.001$).

On the other hand, in none of the problems in the test sections was there a significant difference in completion times between conditions. Particularly worth noting is that results for the Saw Blade problem, since this problem was similar to the Starburst problem, for which there was a large effect of instructional condition. For the Saw Blade problem there was no significant difference conditions ($p = 0.171$) for all participants. It is interesting to note that the median times for the high cognitive load condition, 264.5s (4m:24s), and the low cognitive load conditions, 164.5 (2m:44s), were both close to the low cognitive load times for Starburst, so it appears the participants, regardless of condition, learned something that allowed them to complete Saw Blade quickly, without the help provided to the low cognitive load group that was for Starburst. In section 6.3.3.3, learning during the Starburst problem is examined in more detail.

Table 76 – Summary of differences in times for the problems between conditions.

A green cell indicates that the median time for the low cognitive load group was shorter and red indicates the high cognitive load group had a shorter median time. A saturated colour indicates a significant difference in times between the conditions.

	Time Difference Between Conditions		
Experience Level	All	Self-Assessment ≤ 1	Self-Assessment > 1
Subprograms Instruction	🟢	🟩	🟢
Square	🟢	🟩	🟢
Pentagon	🟩	🟩	🟢
Radiation	🟩	🟩	🟢
H	🟥	🟥	🟥
Bow Tie	🟢	🟢	🟢
House	🟢	🟢	🟢
Subprograms Test	🟢	🟢	🟢
Bow Tie Two	🟢	🟢	🟢
K	🟢	🟢	🟢
Up Arrow	🔴	🔴	🔴
Star	🟢	🟢	🟢
Blade	🔴	🔴	🔴
Loop Instruction	🟩	🟩	🟢
New Square	🟢	🟢	🟢
Radioactive 2	🟢	🟢	🟢
Wall	🟢	🟢	🟢
Circle	🟢	🟢	🟢
Starburst	🟩	🟩	🟩
Rep Test	🟢	🟢	🟢
Hexagon	🔴	🔴	🔴
Bumps	🟢	🟢	🟢
Saw Blade	🟢	🟢	🟢
Gear	🟢	🟢	🟢
Wave	🟢	🟢	🔴
Tree	🟢	🟢	🔴

The above results for timings and success rates will be discussed further in Section 6.3. In addition to the timings and success rates, results concerning the participants' use of various Logo capabilities were also collected and are presented in the next section.

6.3.3 Strategy Use

In the problems in the instructional material, the participants were supposed to learn about rules that permitted problem strategies related to subprograms and loops, concepts identified in the WDA and CTA, and tested in the tests. In this section, results related to the participants' learning and use of strategies related to subprograms and loops are presented. Following those results are a detailed analysis of the Starburst and Saw Blade problems, as those problems were similar and present an opportunity to examine the transfer of strategies related to angle adjustments that was previously described using the CTA decision ladder for Logo.

6.3.3.1 Subprogram Use

One concept identified in the WDA abstraction hierarchy of Figure 16 that the participants were to learn was the use of subprograms. Subprograms, often referred to as procedures or functions, are a very important part of any programming language, so teaching the concept is important. Several of the problems in the subprograms instructions described in Table 15 were designed to teach the participants to develop rules for recognizing when subprograms could be used. This section presents results of how frequently participants made use of subprograms in some of the testing problems, as an indicator of if the participants learned when to use subprograms.

There were three problems in the subprograms test that tested the use of subprograms: Bow Tie Two, Star, and Blade. The other problems in the test tested program creation, but not the use of subprograms. Additionally, the final problem in the subprograms instruction, House, for which the instruction was identical for each instructional condition, and for which little guidance was provided, can be analysed as a test of ability to use subprograms as well.

For each of the programs the number of participants who used subprograms is given. The time used by participants who did and did not use subprograms is also given as an indicator if the use of subprograms reduced work. For the Bow Tie Two problem, because this problem was identical to the Bow Tie problem, it is also investigated if there were any effect of condition on the transfer of subprogram use from Bow Tie to Bow Tie Two.

6.3.3.1.1 House

The first test of use of subprograms was the House problem, even though this problem was in the instructional material. In both instructional conditions, the participants were asked to

"Create a new program called House. Create the following shape (Figure 21). All lines are length 90. Try reusing existing programs."

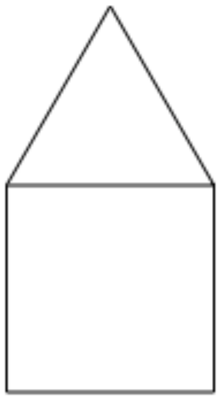


Figure 21 - Figure to be drawn for the House problem

For this problem, the suggestion to reuse existing programs was made. When this problem was created, it was intended that participants would use both Square and Triangle to complete the problem. However, there were a number of ways to complete the program. Some participants reused triangle, some reused both Square and triangle, and despite a suggestion to the contrary, some reused neither (no participants reused only Square). The raw data for each participant can be found in Appendix U. As a reminder, all participants successfully completed this problem.

Table 77 shows the number of participants who used no subprograms, just triangle, or both triangle and Square in their solution to the House problem. As can be seen, all but two participants used at least one subprogram in their solution. Fisher exact tests revealed no dependence of experience level or instructional condition on the number of subprograms used (Experience level: p = 0.8; Instructional Condition: p = 0.613). As can be seen, 37 of the 47 participants used two subprograms and eight used one subprogram, Triangle (since no participants only used Square).

Table 77 – Number of participants who used various subprograms in solution to House problem by experience and instructional condition

	Total Participants	No Subprogram	Triangle Only	Triangle and Square
All	47	2	8	37
Self-Assessment ≤ 1	36	2	7	27
Self-Assessment > 1	11	0	1	10
High Cognitive Load	24	2	4	18
Low Cognitive Load	23	0	4	19

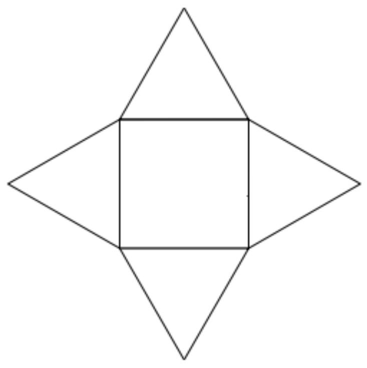
The participants who used no subprograms took a median time of 204s (3m:24s), and the participants who used at least one subprogram took a median time of 84s. A Mann-Whitney U tests did revealed a marginally significant difference (U = 85, p = 0.054). Of those who used both subprograms, the median time was 60s and those who did not use both (none or triangle only) had a median time of 125.5s (2m:5s). The difference was significant using a Mann-Whitney U test (U = 95, $n_{\text{both}} = 37$, $n_{\text{none or triangle only}} = 10$, p = 0.017).

In the House problem, almost all participants used at least one subprogram, as would be expected, since they were told to do so. Most participants used two subprograms, and those participants completed the problem faster than the other participants.

6.3.3.1.2 Star

The second problem that involved testing of the use of subprograms was the Star problem. This problem was in the subprograms test, so the problem statement was the same for both instructional conditions. For this problem participants were asked to

"Create a new program called *Star*. Draw the following image. All lines are length 90."



When completing this problem, some participants used a combination of Square and Triangle. In some cases participants used Triangle one, two, or four times. When creating this problem, I intended that the participants would use the triangle program four times, and adjust the positions of the triangles to create the figure. Some participants also used the Square subprogram to draw the centre of the figure, though that step was not necessary. The data for subprogram use by each participant can be found in Appendix V.

Table 78 summarizes the number of participants who used a number of instances of Triangle in their solution. No participant used three triangles. Two Fisher exact tests revealed no significant dependence of either experience level or instructional condition on triangle use (experience: $p = 0.137$; instructional condition: $p = 0.695$). A significant proportion of the participants used at least one triangle in their solution ($p < 0.001$), and there was a marginally significant proportion of participants who used four triangles ($p = 0.072$).

Table 78 – Number of participant using Triangle in Star solution by experience level and instructional condition (for participants who were successful at Star)

	Total Participants	No Triangle	One or Two Triangles	Four Triangles
All	45	9	7	29
Self-Assessment ≤ 1	34	8	7	19
Self-Assessment > 1	11	1	0	10
High Cognitive Load	23	6	4	13
Low Cognitive Load	22	3	3	16

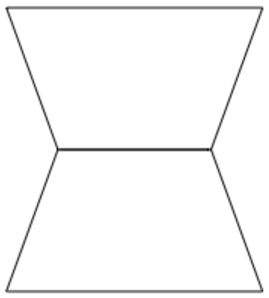
The participants who used at least one triangle subprogram had a median time of 255.5s and those who used no triangle subprograms had a median time of 306s, but a Mann-Whitney U test did not indicate a significant difference (p = 0.1). Of those who used four triangles, the median time was 234s (3m:54s) and those who did not use four triangles (none or one or two) had a median time of 334s (5m:34s). The difference was significant using a Mann-Whitney U test (U = 90, $n_{\text{four}} = 29$, $n_{\text{not four}} = 16$, p < 0.001).

In the Star problem, most participants used four Triangles in completing the problem, and the participants who four triangles completed the problem in significantly less time than the other participants, but the use of subprograms alone did not have a significant effect on completion time.

6.3.3.1.3 Blade

For the Blade problem in the subprogram test, the participants were asked to

> "Draw the following image by creating one program (*Blade one*) to draw the top half and then a second program (*Blade two*) to draw the whole image. The short horizontal line and the diagonal lines are 100 pixels. The long horizontal lines are 170 pixels."



In the problem analysis in section 0, the success rates and times for the whole Blade problem, including both Blade One and Blade Two, were examined. This section presents results related to the reused of Blade One in completing Blade Two and how reuse affected completion time. It was expected that the participants would use the Blade One program twice when completing Blade Two; however, the simplicity with which that solution was implemented depended on the order in which the lines were drawn in Blade One, so not all participants were able to easily reuse Blade One twice. Some reused Blade One once, and then drew the other half from scratch, and, finally, some participants did not reused Blade One at all in their solution to Blade Two. The data for subprogram use in Blade Two for each participant can be found in Appendix X. In this section, after some preliminary analysis to determine which participants generated appropriate data to study, the reuse of Blade One in the solution of Blade Two is examined.

First, how success in Blade One and Blade Two were related will be examined. Six participants were not successful at Blade One. Two participants did not attempt Blade Two after failing to complete Blade One. Of the remaining four who did not complete Blade Two successfully, all were in the low experience group and three were in the high cognitive load group**.** Table 79 shows the relation of number of people who were successful at Blade One and Two.

Table 79 – Number of successes at Blade One and Blade Two

	Blade Two No Attempt	Blade Two Success	Blade Two Failure
Blade One Success	0	38	3
Blade One Failure	2	0	4

The examination of subprogram use in Blade Two will focus on the 38 participants who successfully completed both Blade One and Blade Two and their use of Blade One, either once, twice, or not at all, and how that use was related to experience or instructional condition and affected completion times will be examined.

Table 80 shows the number of participants who used Blade One a particular number of times in Blade Two. Two Fisher exact tests did not reveal a significant dependence of experience or instructional condition on the number of times Blade One was used (experience: $p = 0.391$, instructional condition: $p = 1$).

Table 80 - Number of participants who used Blade One a particular number of times in Blade Two, by experience and condition

	Total Participants	No Blade One	One Blade One	Two Blade One
All	38	4	5	29
Self-Assessment ≤ 1	28	3	5	20
Self-Assessment > 1	10	1	0	9
High Cognitive Load	18	2	2	14
Low Cognitive Load	20	2	3	15

Participants who used no instances of Blade One, one instance of Blade One, and two instances of Blade Two completed Blade Two with median times of 267.5s, 282s, and 195s respectively. A Kruskal-Wallis rank sum test did not reveal a difference between at least two of the time distributions ($X^2 = 2.511, \mathrm{df} = 2, \mathrm{p} = 0.285$). Of those who used Blade Two twice, the median time was 195s (3m:15s) and those who did not use Blade Two twice had a median time of 282 (4m:42s). The difference was not shown to be significant using a Mann-Whitney U test ($U = 92.5$, $n_{\mathrm{twice}} = 29$, $n_{\mathrm{not\ twice}} = 9$, $p < 0.198$).

In the Blade problem, most participants used Blade One twice to solve the Blade Two sub problem, though using that strategy did not give a significant time advantage.

6.3.3.1.4 Bow Tie and Bow Tie Two

Bow Tie, found in the sub programs instructional material, and Bow Tie Two, found in the sub program test, required the participants to create the same final program, shown in Figure 22.

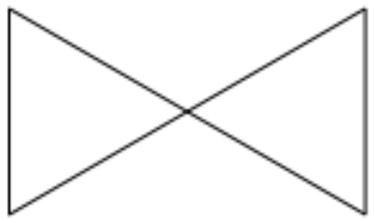


Figure 22 - Bow Tie

For Bow Tie, in the high cognitive load instructional material, the participants were asked to

"Create a program called *Bow Tie* that draws the following figure (see Figure 22). The triangle sides are all length 90."

The participants in the low cognitive load condition were given a number of steps that they had to complete. First they were asked to add the lines

Right 0

My Triangle

Then they were asked to adjust the orientation of the triangle as in Figure 23

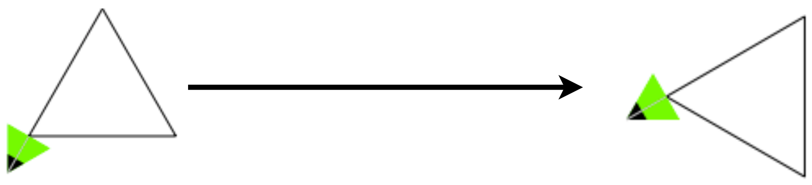


Figure 23 - Adjustment of first triangle added in Bow Tie

Then they were asked to add a second set of the lines

Right 0

My Triangle

and loop the adjustment process to complete the final figure (see Figure 22).

For Bow Tie Two, in the sub programs test, all participants were asked to

"Create a program called *Bow Tie Two* that draws the following figure (see Figure 22). The triangle sides are all length 90. Please do not look at your previous code or the instructional material."

The Bow Tie program was designed with the intention that the participants would complete the problem by using Triangle as a subprogram. This section presents the results on the use of the Triangle subprogram in Bow Tie and Bow Tie Two. Appendix T shows which participants used the Triangle program as part of their solution for Bow Tie and Bow Tie Two.

Table 81 shows the number of participants in each condition who did or did not use subprograms in their solution to Bow Tie by experience and instructional condition, for those successful at Bow Tie. The instructions for the low cognitive load condition guided the participants through a process of using the Triangle subprogram, so it could be expected that all those participants would use Triangle. A Fisher exact test did not reveal an effect of experience on subprogram use ($p = 0.457$). As expected, all participants in the low cognitive load condition used the Triangle subprogram; however, fewer than half those in the high cognitive load condition used it (9 of 21). A Fisher exact test showed a significant effect of instructional condition on subprogram usage ($p < 0.001$).

Table 81 - Number of participants who used Triangle in their solution to Bow Tie by experience and instructional condition, for those successful at Bow Tie

	Total Participants	Triangle Subprogram	No Triangle Subprogram
All	41	29	12
Self-Assessment ≤ 1	30	20	10
Self-Assessment > 1	11	9	2
High Cognitive Load	21	9	12
Low Cognitive Load	20	20	0

Of those successful at Bow Tie, the median time for those who used the Triangle subprogram was 135s and 215.5s for those who did not use it. A Mann-Whitney U test did not reveal a significant difference between the median times ($U = 121.5$, $n_{\text{subprogram}} = 29, n_{\text{no subprogram}} = 12$, $p = 0.136$).

The same results are now presented for the Bow Tie Two problem. For this problem, as previously mentioned, four participants were counted as not attempting the problem: two reused Bow Tie to complete Bow Tie Two, as they were asked not to do, and two did not attempt the problem at all.

Table 82 shows the number of participants in each condition that did or did not use subprograms in their solution to Bow Tie Two by experience and instructional condition, for those successful at Bow Tie Two. A Fisher exact test did not reveal a significant effect of experience level on subprogram use (p = 0.476); however, a significant effect of condition on subprogram use was detected (p = 0.008).

Table 82 - Number of participants who used the Triangle subprogram in their solution to Bow Tie Two by experience and instructional condition, for those successful at Bow Tie Two

	Total Participants	Triangle Subprogram	No Triangle Subprogram
All	37	19	18
Self-Assessment ≤ 1	29	12	14
Self-Assessment > 1	11	7	4
High Cognitive Load	18	5	13
Low Cognitive Load	19	14	5

Of those successful at Bow Tie Two, the median time for those who used the Triangle subprogram was 129s and 141s for those who did not use it. A Mann-Whitney U test did not reveal a significant difference between the median times (U = 150, $n_{\text{subprogram}} = 19, n_{\text{no subprogram}} = 18$, p = 0.533). Only about half the participants used subprograms for Bow Tie Two. It is possible that the use of subprograms in Bow Tie was related to the use of subprograms in Bow Tie Two.

Table 83 shows the number of participants who used subprograms in Bow Tie and Bow Tie Two. First, a Fisher exact test shows that for all participants, the use of the Triangle subprogram in Bow Tie Two was dependent on Triangle subprogram use in Bow Tie (p = 0.017), indicating a relationship between subprogram use in Bow Tie and Bow Tie Two.

Table 83 – Number of participants who used subprograms in Bow Tie and Bow Tie Two, for participants successful at both problems

		Bow Tie Two Subprogram	No Bow Tie Two Subprogram
All	Bow Tie Subprogram	17	10
	No Bow Tie Subprogram	1	7

In the above sections, the use of subprograms was examined. In section 6.3.3.4, the results are summarized along with other strategies analyses involving loops and transfer from Starburst to Saw Blade, which are presented in the next two sections.

6.3.3.2 Loop Use

Another Logo feature identified in the WDA abstraction hierarchy was loops. The problems in the loop instructional materials were designed to help the participants learn rules to identify when loops might be useful in the completion of programs and learn how to use loops. The problems in the loop test examined the participants' use of loops to solve problems. There were several problems for which participants had to use the repeat command to create loops to avoid using a large number of commands, such as Starburst, Gear, Wave, Circle, and Saw Blade. Without using a loop, those problems would be unmanageable, as without a loop, a solution would require hundreds of commands (for Gear at least 360 commands would be needed). There were three problems, Hexagon, Bumps and Tree that were designed to test the participants' use of loops, but were manageable without using loops. In order to examine if participants learned to identify when loops were useful, the number of participants who did and did not use loops in those problems is examined along with the effect of using loops on completion time for those problems.

6.3.3.2.1 Hexagon

The Hexagon problem was the first problem in the loops test. For this problem, the participants were asked to

"Create a program called *Hexagon*. Create a six-sided figure where each line is length 50 and each angle is a left turn of 60."

To solve this problem, a participant could add the lines

```
Forward 50
Left 60
```

five times or insert the two commands inside a repeat statement.

All participants successfully completed the Hexagon problem. Appendix Y shows which participants used a loop for the Hexagon problem. Only two participants did not use a loop in their solution to the Hexagon problem. The use of a loop by experience and instructional condition is shown in Table 84. Two Fisher exact tests did not reveal a significant effect of experience or instructional condition on loop use (experience: p = 1;instructional condition: p = 1).

Table 84 - Number of participants who did and did not use the Repeat command in Hexagon by experience and instructional condition

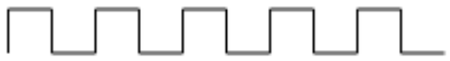
	Total Participants	Used Loops	Did Not Use Loops
All	47	45	2
Self-Assessment ≤ 1	36	34	2
Self-Assessment > 1	11	11	0
High Cognitive Load	24	23	1
Low Cognitive Load	23	22	1

The median time to complete the Hexagon problem by those who used a loop was 58s. For the two who did not use a loop, their median time was 101.5s. A Mann-Whitney U test did not reveal a significant difference between the times (W = 76.5, $n_{\text{repeat}} = 45, n_{\text{no repeat}} = 2$, p = 0.102).

6.3.3.2.2 Bumps

The Bumps problem was the second problem in the loop test. For this problem participants were asked to

"Create the following figure in a new program called *Bumps*. The lines are each 20 pixels long."

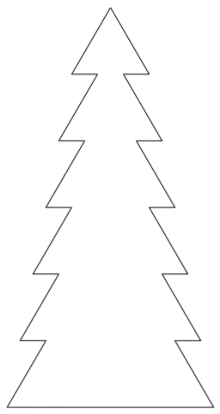


Every one of the 42 participants who successfully completed the Bumps problem used a loop.

6.3.3.2.3 Tree

The last program in the Loop test, Tree, was also a program where participants could use repeat, but could have completed the problem without using it. For this problem, participants were asked to

"Create a program called *Tree* and draw the following figure. The slanted lines are length 90 and the short horizontal lines are length 30."



There are a number of approaches participants could have taken to complete this problem, using various numbers of loops. Appendix Y shows the number of loops used by each participant. One participant used three loops: he drew one side of the tree and then retraced that side using the second loop and then drew the second side using the third loop. There were two participants who did not attempt this problem, as they had given up by this point in the study. Of the 45 participants who attempted the problem, 35 completed the problem successfully.

Table 85 shows the number of participants who used no, one, two, or three loops in their solution to Tree for the participants who completed the problem successfully. Two Fisher exact tests revealed no dependence of loop use on experience or instructional condition (experience: p = 0.787; instructional condition: p = 1).

Table 85 - Number of participants who used various numbers of loops in Tree by experience and instructional condition for participants successful at Tree

	Total Participants	No Loops	One Loop	Two Loops	Three Loops
All	35	1	2	31	1
Self-Assessment ≤ 1	24	1	2	20	1
Self-Assessment > 1	11	0	0	11	0
High Cognitive Load	18	1	1	16	0
Low Cognitive Load	17	0	1	15	1

The median time taken to complete the Tree problem by the participants who used no, one, two or three loops was 392s, 1057s, 601s, 861s. A Kruskal-Wallis test revealed no significant difference between and of the times for the difference number of loops used ($X^2 = 4.332, \mathrm{df} = 3, \mathrm{p} = 0.227$).

The above results examine how many participants used loops in some test programs. The results will be discussed in section 6.4.2. In the next section, the rules that allow small adjustments to solve more complex problems will be examined.

6.3.3.3 Transfer from Starburst to Saw Blade

Two problems, Starburst, from the loop instructions, and Saw Blade, from the loop test, were closely related problems. A solution to either problem required finding the proper angles between the line segments in the figure, and, hence, possibly required the fine-tuning of angles; therefore, the development of rules for the appropriate angles was identified as a goal of the training material for

Starburst. In this section the two problems are reviewed, then the determination of angles to solve the problem is examined, and then possible transfer between the two problems as related to the ability of the participants to pick the correct angles for solving the problems is explored. Participant 35 is excluded from the analysis for these problems because she did not attempt any problems past Bumps in the loop test.

6.3.3.3.1 The Problems

The Starburst problem, in the loop instructions, asked participants to

"Create a program called *Starburst.* Create the following figure (Figure 24). The lines are 50 pixels long. There are 36 outward points."

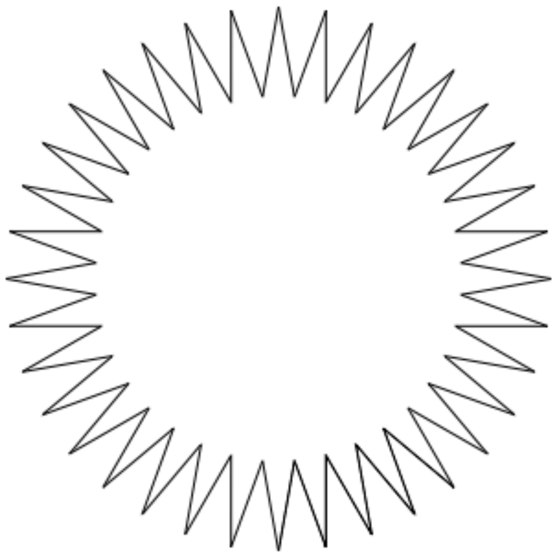
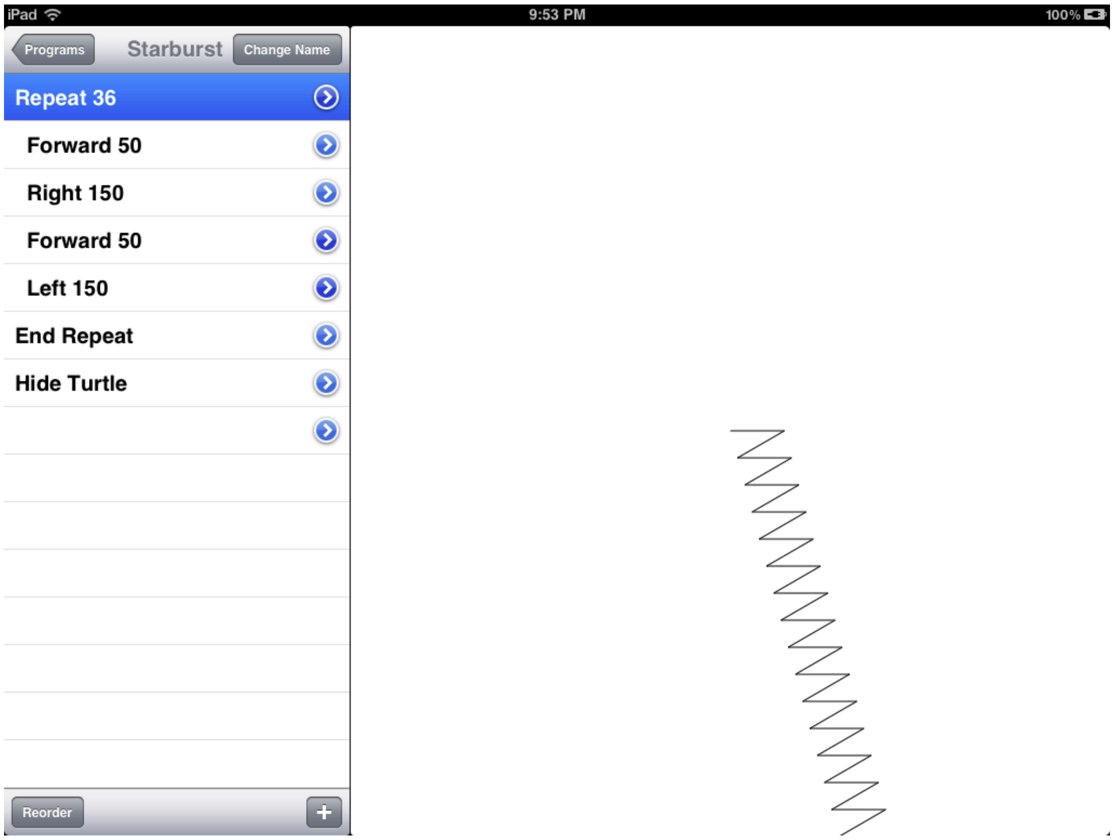


Figure 24 - Starburst figure

Participants in the high cognitive load condition were given the length of the lines and the number of points. In the low cognitive load condition, the participants were given a number of steps to achieve the intermediate program and figure shown in Figure 25. The participants were then told to adjust the angles to achieve the circular shape of the final Starburst shape in Figure 24.



Repeat 36
Forward 50
Right 150
Forward 50
Left 150
End Repeat
Hide Turtle

Figure 25 - Intermediate step given to low cognitive load condition participants for Starburst problem

For the Saw Blade problem, the participants in both conditions were asked to

"Create a program calls *Saw Blade.* Draw the following figure (Figure 26). There are 36 teeth. Long lines are 50 pixels long; short lines are 30 pixels long."

Because the Saw Blade problem was part of the loop test, the participants, in both conditions, were not given any further guidance.

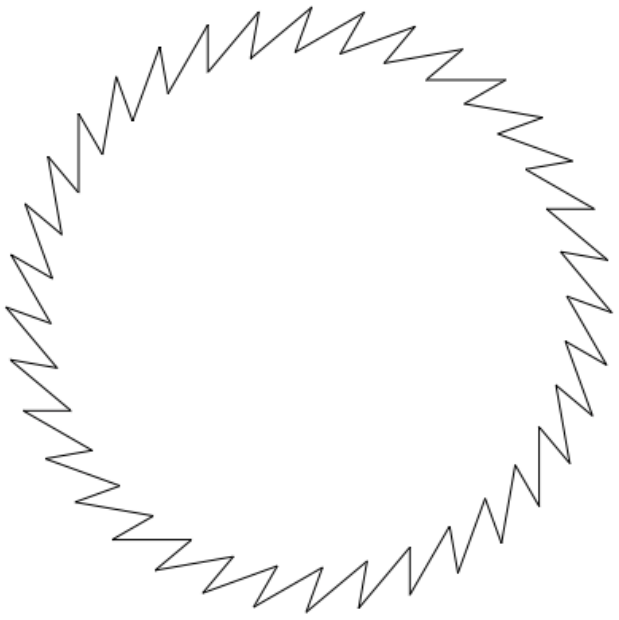


Figure 26 - Saw Blade

The possibility of participants transferring knowledge from the Starburst problem to the Saw Blade problem existed because the problems were closely related. Figure 24 and Figure 26 show the visual similarity of the two problem solutions, and Table 86 shows how similar solutions can be used for the two problems.

Table 86 - Solutions for Starburst and Saw Blade problems

Example Starburst Program	Example Saw Blade Program

`Repeat 36`
`    Forward 50`
`    Right 150`
`    Forward 50`
`    Left 160`
`End Repeat`
`Hide Turtle`
	`Repeat 36`
	`    Forward 50`
	`    Right 150`
	`    Forward 30`
	`    Left 160`
	`End Repeat`
	`Hide Turtle`

6.3.3.3.2 Angle Choice and the Decision Ladder

In both the Starburst and Saw Blade problems, the participants needed to choose the correct angles between the line segments. In the decision ladder found in Figure 17 and the discussion that followed, it was noted that participants could potentially cycle around an alert-procedure-execute or alert-task-procedure-execute cycle, without considering the rest of the decision ladder states, in order to make small adjustments to an angle. It would be beneficial, as was determined in the development of rules for appropriate context, for a participant to learn what angle, or range of angles, to use in a problem such as Saw Blade or Starburst, as they could then complete such an adjustment task more quickly and with less effort. In the remainder of this section, the angle adjustments used in Starburst and Saw Blade will be examined to determine if some transfer between the problems occurred.

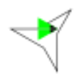
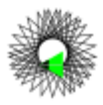
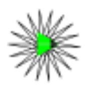
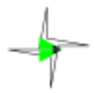
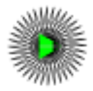
6.3.3.3.3 Completion Times

The participants who succeeded at Starburst took median times of 578s (9m:38s) and 224.5s (3m:44s) for the high and low conditions respectively. The participants who were successful at the Saw Blade problem took median times of 264.5s (4m:24s) and 164.5s (2m:44s) for the high and low conditions respectively. If only the participants who successfully completed both problems are considered, the median completion times for the high cognitive load participants went from 578s (9m:38s) for Starburst to 247.5s (4m:7s) for Saw Blade, a significant decrease as shown with a Wilcoxon Signed-Rank test ($W = 50$, $n = 16$, $p = 0.003$). Similarly, for the participants in the low cognitive condition, the median completion times went from 217.5s (3m:37s) for Starburst to 161s (2m:41s) for Saw Blade, a non-significant decrease ($W = 85$, $n = 16$, $p = 0.11$). It is interesting to note that despite none of the participants having guidance in solving the Saw Blade problem, participants in both conditions had median times close to that of the participants in the low cognitive load condition for Starburst. Therefore, it seems that participants in both conditions learned something that helped with their solutions to Saw Blade.

6.3.3.3.4 Number of Angle Adjustments

As just seen, the participants used less time to complete Saw Blade compared to Starburst. One source of time spent in completing the Starburst and Saw Blade problems was adjusting angles. The subtask of adjusting the angles properly was not a trivial one for some participants, as they did not know how much to adjust the angles by. As a result, participants, particularly those in the high cognitive load group, had to make a lot of angle adjustments when completing Starburst. For example, Table 87 shows a subsequence from participant 26's solution where she is adjusting the angles in her solution. As a result of even small adjustments, the resulting figure can change quite significantly, potentially making it difficult for a participant to determine what range the angles should be in. It can be determined if there was a relationship between the number of angle adjustments and the time reduction between the problems.

Table 87 - Sequence of angle adjustments in participant 26's solution

```
Repeat 36     Repeat 36     Repeat 36     Repeat 36     Repeat 36
  Left 30       Left 30       Left 30       Left 80       Left 60
  Forward 50    Forward 50    Forward 50    Forward 50    Forward 50
  Right 150     Right 145     Right 170     Right 170     Right 170
  Forward 36    Forward 36    Forward 36    Forward 36    Forward 36
End Repeat    End Repeat    End Repeat    End Repeat    End Repeat
```

Appendix AA shows the number of angle adjustments that all the participants had to make for the Starburst and Saw Blade problems. For this analysis, only participants who were successful at both the Starburst and Saw Blade problems are considered, in order to have comparable data. I will look at the difference in timings and angle changes between those two problems. Table 88 shows the median number of angle changes for the problems, and the median difference between the problems. There was a significant difference in the number of angle adjustments made from the Starburst to Saw Blade problems for both instructional conditions (high: W=27, n = 16, p = 0.036; low: W = 28, n = 16, p = 0.041). Furthermore, there was a significant difference in the number of angle changes between conditions for both the Starburst and Saw Blade problems, as shown with a Mann-Whitney U test (Starburst: U = 60.5, $n_{low} = 16, n_{high} = 16, p = 0.011$, Saw Blade: U = 75.5, $n_{low} = 16, n_{high} = 16, p = 0.0490$).

Table 88 – Median number of angle changes by condition and problem

Cognitive Load Condition	Number of Starburst Angle Changes	Number of Saw Blade Angle Changes	Median Difference
High	22	8.5	17.5
Low	7.5	3	4

As discussed, there was a large drop in the time taken from the Starburst problem to the Saw Blade problem. To examine this relationship between median number of angle changes and problem solution time, the change in problem time data and change in angle adjustment data for the high and low cognitive load groups for participants who successfully completed both problems was fit to the following model:

Equation 3 – Relation between time difference from Starburst to Saw Blade and changes in number of angle adjustments

$$\begin{aligned}\mathbf{Saw\ Blade\ Time - Starburst\ Time} \\ = \mathbf{b_0 + b_1(Saw\ Blade\ Angle\ Adjustments - Starburst\ Angle\ Adjustments)}\end{aligned}$$

Table 89 - Coefficients and p-values for model in Equation 3 for participants who were successful at both problems. The low cognitive load model is not valid due to a violation of a normality of residuals assumption.

	High Cognitive Load		Low Cognitive Load	
b_0	-147.00	(0.109)	68.844	(0.158)
b_1	26.196	(<0.001)*	11.585	(0.005)*
R^2	0.821	(<0.001)*	.442	(0.005)*

From Table 89, it can be seen that the change in number of angle adjustments had a high relationship (82.1% for the high cognitive load and 44.2% for the low cognitive load) to the change in time. However, the normality assumption for the residuals in the low cognitive load condition was not met, so the estimate for R^2 may not be significantly different from 0.

The number of angle adjustments decreased from Starburst to Saw Blade, but it is still not certain if the participants were able to pick angles better in Saw Blade. To examine this issue, the range of angles used can be checked. Appendix AA shows the maximum, and minimum, of the angles that each participant used. Table 90 and Table 91 summarize the data, showing the medians, for each measure indicated. Wilcoxon signed-rank tests were performed for the range of angles that the participants tried.

Table 90 - Median of the minimum and maximum angles and range of angles tried by participants in the high cognitive load group who where successful at both Starburst and Saw Blade

	Minimum Angle	Maximum Angle	Angle Range
Starburst	1	200	199
Saw Blade	31.5	175	137

Table 91 - Median of the minimum and maximum angles and range of angles tried by participants in the low cognitive load group who where successful at both Starburst and Saw Blade

	Minimum Angle	Maximum Angle	Angle Range
Starburst	115	180	55
Saw Blade	135	160	25

The results summarized in Table 90 show that the high cognitive load participants were able to work within a significantly smaller range (W = 17, n = 16, p = 0.009) of angles when completing the Saw Blade problem compared to the Starburst problem, a fact that probably helped with the reduction in number of angle adjustments and, hence, the time needed to complete Saw Blade. The results summarized in Table 91 do not show that there was a significant difference (W = 90, n = 16, p = 0.083) in angle range used by the low cognitive load participants between Starburst and Saw Blade. However, when completing the Saw Blade problem, the participants in the low cognitive load group still worked with in a smaller range of angles than those in the high cognitive load condition (significant using a Mann-Whitney U test with W=50, $n_{low} = 16, n_{high} = 16$, p=0.003), indicating that the participants in the low cognitive load group had better learned what angles to use than those in the high cognitive load group.

The above results for Starburst and Saw Blade indicate that the students learned something during the study that helped improve their results between the two problems. In the discussion, how the learning of angles ranges was affected by cognitive load control will be examined in terms of behaviours described in the decision ladder.

6.3.3.4 Summary of Strategy Analysis

In this section, results relevant to various potential strategies and rules, which were identified by examining the WDA and CTA for Logo, are presented. In section 6.3.3.1, the use of subprograms was examined. In all problems analysed for subprogram use, most of the participants used the subprograms. In the House problem, 37 of the 47 participants used both Square and Triangle as subprograms in their solution, and the participants who used both subprograms had a significantly lower median completion time of 60s compared to the participants who did not use both subprograms with a median time of 125.5s. For the Star problem 37 of the 45 participants who successfully completed the problem used subprograms. For the Blade problem, 29 of the 38 participants who successfully completed the problem used two copies of Blade One in their solution to Blade Two, though those using Blade Two twice did not have a significantly different completion time from the participants who did not use it twice. Finally, in Bow Tie Two, 19 of the 37 participants who succeeded at the problem used the Triangle subprogram.

For none of the problems was there a significant effect of experience level on use of subprograms. Only the Bow Tie Two problem led to a significant effect of instructional condition on the use of subprograms, with more participants in the low cognitive load condition using subprograms than in the high cognitive load condition.

After the results of subprogram use, data on the use of loops was presented in section 6.3.3.2, for three problems in the loop test that were possible to do in a reasonable time without a loop. For the Hexagon problem, 45 of the participants used a loop and 2 did not. For the Wall problem, all participants used a loop. For the Tree problem, 31 of the 35 participants successful at the problem used two loops.

Finally, in section 6.3.3.3, data on the effect of instructional condition on how the participants selected angles in the Starburst and Saw Blade problems was presented. There was a significant drop in angles range used by the participants in the high cognitive load condition between the problems, but the low cognitive load condition participants still used a significantly smaller range of angles when completing Saw Blade, indicating they had better learned what range of angles to use.

Before discussing the above results and the consequences of them, the NASA-TLX results for the study will be presented.

6.3.4 NASA-TLX Results

As an assessment of workload experienced during the study, the participants completed the NASA-TLX five times during the study after each of the introduction, subprogram instruction, subprogram test, loop instruction, and loop test.

All the NASA-TLX scores for all participants can be found in Appendix BB. Two participants are excluded from this analysis because of missing NASA-TLX results. Participant 35 did not complete enough of the loop test to make the NASA-TLX result meaningful; therefore, that participant is also excluded from the NASA-TLX analysis. For each test there are 22 results in each of the low and high cognitive load conditions.

Table 92 shows the NASA-TLX results (weighted averages) for each of the study sections. For none of the sections was a significant difference in TLX result between condition detected using a series of Mann-Whitney U tests (intro: U = 245, p = 0.787; subprogram instruction: U = 263, p = 0.634; subprogram test: U = 239, p = 0.953; loop instruction: U = 162, p = 0.062; loop test: U = 217.5, p = 0.573). There was nearly a significant result for the loop instruction, which was the section of the study that showed the most difference in completion time between instruction conditions.

Table 92 – NASA-TLX Weighted Averages for study sections by instructional condition

	Intro	Subprogram Instruction	Subprogram Test	Loop Instruction	Loop Test
all	3.625	4.12	5.475	4.69	5.98
High Cognitive Load	3.185	4.04	5.415	5.49	5.95
Low Cognitive Load	3.945	4.2	5.52	4.25	5.98

It is possible that there is a significant difference in the NASA-TLX results that is hidden by the individual differences in interpretation of the NASA-TLX and mapping of task difficulty to the TLX scale. Using the TLX results for the introduction as a control, since those results measured a common task, the following model can be used to look for a significant effect of condition on TLX results:

$$\text{TLX Section Result} = b_0 + b_1\text{Intro TLX Result} + b_2\text{ConditionHigh}.$$

Table 93 shows the coefficients for the above model for each section. R^2 is based on an adjusted correlation coefficient. For none of the TLX results was there a significant interaction effect between the introductory material TLX result and condition.

Table 93 – Coefficients and R^2 for TLX Model for each section (p values given in parenthesis)

Section Experience Measure	Subprogram Instruction	Subprogram Test	Loop Instruction	Loop Test
b_0 (intercept)	1.453 (0.001)	2.773 (<0.001)	3.014 (<0.001)	3.979 (<0.001)
b_1 (intro TLX)	0.717 (<0.001)	0.673 (<0.001)	0.695 (<0.001)	0.549 (0.002)
b_2 (condition)	-0.128 (0.129)	-0.055 (0.913)	1.181 (0.039)	0.288 (0.631)
R^2	0.542 (<0.001)	0.326 (<0.001)	0.345 (<0.001)	0.170 (0.008)

The only section for which there was a significant effect of condition on the TLX score was for the loop instruction section. The residuals for the model for loop instructions were sufficiently normal, as shown with a Shapiro-wilk test (W = 0.969, n = 44, p = 0.292), and there was a uniformity of variance in the residuals with no obvious patterns.

6.4 Discussion

In this study, the effect of two different sets of instructional materials on the learning and transfer of a set of rules was tested. The set of rules was developed from a WDA and CTA and based on the guidelines in Chapter 4. The two sets of instructional materials were designed to impose different levels of cognitive load and they defined the high and low cognitive conditions used in this study.

There were six sections of this study: experience assessment survey, introduction instruction, subprograms instruction, subprograms test, loop instruction, and loop test. For the subprograms and loop instructions, different types of instructional methods were used for the two conditions; however, the introductory instructions and the tests were the same for the two conditions. After each instructional and testing step, the participants completed the NASA-TLX as a measure of their workload during the particular study section.

This study gave the opportunity to test the hypothesis that the participants in the low cognitive load condition would complete the instructional material faster, perform better on the transfer tests in terms of success rate and completion time, and experience a lower workload during learning and testing. Additionally, the study allowed the testing of rule transfer for rules identified from the decision ladder found in Figure 17.

In this section, the effects of the two sets of instructional materials on learning of rules is discussed through the effects on timings, successes, and workload, followed by the effects on transfer of rules.

6.4.1 Times, Successes, and Workload

In this section the effect of the two sets of instructional materials are discussed in terms of timings, successes, and workload. Some of the effects will be discussed in terms of experience levels, which were assessed in the survey given to the participants at the start of the study, which collected a number of measures of experience. As discussed in section 6.3.1, the participants were divided into two experience levels, those with self-assessment ≤ 1 and those with self-assessment ≥ 1.

6.4.1.1 Section-Level Results

In the analysis of the study, results were tabulated at both the study section level and the problem level. For neither of the instructional sections was there a statistically significant difference in success rates between conditions. However, there were differences in time performance for the instructional sections. There was a significant difference in time spent on the subprograms instruction for those with a self-assessment ≤ 1, with the high cognitive load condition participants having a median time of 16m:8s and the low cognitive load condition participants having a median time of 13m:45s, a savings of 2m:23s. For the loop instructions there was a significant difference in completion time for all participants, with the low cognitive load participants having a median time of 22m:50s and those in the high cognitive load condition having a median time of 13m:45s, a difference of 9m:05s. For the participants with self-assessment ≤ 1, the difference between conditions for the loop instruction was larger, with the low cognitive load participants having a median time of 24m:27s and those in the high cognitive load condition having a median time of 14m:6s, a difference of 10m:21s.

The decrease in benefit between instructional conditions based on experience level is consistent with the guidance fading effect from CLT, which predicts that cognitive supports may become less effective for more highly experienced individuals. Furthermore, the lack of significant difference in instructional condition when the participants with self-assessment >1 are included is further evidence that with the easier problems found in the subprograms instruction, higher experience individuals do not benefit from the additional cognitive support.

Despite the difference in timing between conditions for the instructional material, there was no significant effect of condition on success rates or times for the tests. It was hypothesized that the participants in the low cognitive load condition would complete the tests faster, having learned more during the instructional stages, but this result of no significant difference is still interesting. The participants in the low cognitive load condition spent less time on the loop instructions, and experienced a lower workload, but did not spend a significantly different amount of time on the test, nor experience significantly different workload, than the high cognitive load participants. This result suggests that, even with taking less time on instruction, the low cognitive load condition participants learned a sufficient amount to complete the problems in the test as quickly and with as many successes as those in the high cognitive load condition. The same situation occurred for the participants with self-assessment ≤ 1, for subprograms, except for a significant difference of workload.

6.4.1.2 Individual Problems

In addition to an analysis at the section level, results were also gathered for the individual problems. For some of the early problems in the study there was a significant difference in completion time. For the Radiation and Pentagon problems, the low cognitive load participants completed the problems faster. For the Radiation problem, the high cognitive load participants took a median time of 48.5s and the low cognitive load participants took a median time of 40s. For the participants with self-assessment ≤ 1, the difference was larger: 59s vs 42s. For the Pentagon problem, the high cognitive load participants took a median time of 1m:36s and the low cognitive load participants took a median time of 1m:17s. For the participants with self-assessment ≤ 1, the difference for Pentagon was marginally larger: 1m:36s vs 1m:15s. After those problems, there was no significant difference in time for any problem until Starburst. For the Starburst problem, the high cognitive load participants took a median time of 9m:32s and the low cognitive load participants took a median time of 3m:44s, a difference of 5m:48s. For the participants with self-assessment ≤ 1, the difference was larger: 14m:59s vs 3m:49s, a difference of 11m:10s.

The evidence points to a conclusion that the low cognitive load condition provided a slight advantage as the participants were starting to learn Logo, but, after the radioactive problem, the problems in the instructional material were not sufficiently difficult for the difference in instructional types to provide a significant advantage until the Starburst problem. The Starburst problem was the most difficult problem in the instructional material, so there was greater opportunity for the low cognitive load instructional material to provide a significant cognitive support, and the difference in time shows that low cognitive load instruction did provide some advantage.

As with the testing sections, there were no test problems that showed a significant difference in completion times between conditions. Again, it is noted that, despite the different amounts of time spent on the instruction problems, there was not a significant effect of condition on completion time for any test problem.

These results related to the sections and individual problem success rates and times present evidence of the low cognitive load participants learning equally well as the high cognitive load participants, but doing so more efficiently during the instructional sections of the study, indicated by lower learning times but equal performance in the tests. However, these results do not address the learning of the rules identified in the development of the instructional materials. To partly address that issue, the learning of the rules pertaining to the use of subprograms and loops will be discussed in the next section.

6.4.2 Subprograms and Loops

In the design of the instructional material, rules for identifying the usefulness or need for subprograms and loops, and then how to use them, were identified as an instructional goal. In this section, the use of subprograms and loops, as a potential indicator of the use of those rules, is discussed.

Some of the programs in the testing section were designed to test if participants recognized the usefulness of subprograms and could use them in their solutions. These programs were Star, Blade, and Bow Tie Two. In each of these programs, most of the participants who were successful at the problems used subprograms (Star: 36/45; Blade: 34/38; Bow Tie Two: 19/37), with a significant proportion using subprograms for Star, and Blade, providing some evidence that the participants had learned to recognize when they could use subprograms and successfully use them. The House problem results provide additional evidence that a significant proportion knew how to use subprograms, with 45 out of 47 participants using subprograms, but that problem does not test the recognition for the usefulness of subprograms, because the participants were told two use subprograms for that problem. Except for Bow Tie Two, there was no effect of condition on the use of subprograms for these problems.

The fact that there was not a statistically significant proportion of participants using subprograms in Bow Tie Two could be due to a lack of significant time savings in using subprograms for the problem or due to a transfer effect. There is some evidence for a transfer effect. In Bow Tie, in the instructional material, the participants in the low cognitive load group were shown to use subprograms, and all of them did. In the high cognitive load group, the participants were not shown to use subprograms and 9 of the 21 successful at Bow Tie in the high cognitive load condition used subprograms. Given those levels of subprogram use, there was a significant effect of instructional condition on subprogram use for the Bow Tie problem. It was then shown that the participants' use of subprograms in Bow Tie Two was dependent on their use of subprograms in Bow Tie, with 24 of 35 participants who successfully completed both problems exhibiting the same use, or lack of use, of subprograms in the two problems. Not surprisingly, there was also then a significant dependence of subprogram use on condition for Bow Tie Two, with 5 of 18 in the high cognitive load condition and 14 of 19 in the low cognitive load condition using subprograms. Therefore, with an effect of subprogram use during training in Bow Tie on subprogram use in Bow Tie Two, and an effect of condition on subprogram use in Bow Tie Two, evidence supporting rule transfer was seen in the use of subprograms between the problems.

In addition to subprograms, the recognition and use of loops was identified in the development of rules. Several testing problems were not solvable in a reasonable time without using loops, due to the number of commands that would be needed. However, there were three test problems in which loops could be used, but where the problems could be solved in a reasonable time without using loops: Hexagon, Wall, and Tree. For Hexagon, 45 out of 47 used loops, for Bumps all 42 successful at the problem used loops, and for Tree 34 out of 35 successful at the problem used loops. For all of these problems, a significant proportion of participants used loops as expected, indicating that the participants learned to use loops where appropriate, as identified as a goal rule in the development of the loop instructions. For none of these problems was there an effect of condition on loop use, indicating that participants in both conditions learned to use loops.

The above discussion provides evidence that the participants in the study learned to use subprograms and loop during instruction. Only in the Bow Tie Two problem was there evidence of an effect of condition on the use of subprograms, and, in that case, there was evidence of a transfer effect, possibly due to instructional condition. In addition to the use of subprograms and loops, potential rules related to the adjustment of angles were identified from the Logo decision ladder. In the next section the results related to angle adjustment in two problems, Starburst and Saw Blade, are discussed.

6.4.3 Rules for Adjustment of Angles

In the discussion on the development of the Logo decision ladder, in section 6.2.5, the possibility of a cycle within the decision ladder between the alert, task (or procedure), and execution states and activities when making adjustments to a figure was discussed. It was later noted that participants had to make many angle adjustments in completing both the Starburst and Saw Blade problems. This adjustment behaviour could be explained as a potential cycle within the decision ladder, so this pair of problems can be examined for evidence of learning of the rules for the decision ladder cycles.

Initial evidence of learning between these problems can be seen by considering the completion times for those who completed both problems successfully. For the Starburst problem, in the loop instructional material, the median completion times were 3m:37s for the low cognitive load condition and 9m:38s for the high cognitive load condition. In the high cognitive load condition, the participants had no cognitive support in completing Starburst. However, when completing the similar Saw Blade problem, when no participant had cognitive support, the median completion times were 2m:41s and 4m:7s for the low and high condition respectively. Those in the high cognitive load condition significantly decrease their completion time, and because the Saw Blade times, achieved without cognitive support, were closer to the low cognitive load time for Starburst, there is evidence that participants in both conditions learned something to help them complete Saw Blade faster.

As a possible explanation for the improved performance on Saw Blade, further results showed that the participants significantly reduced the median number of angle adjustments they made from Saw Blade to Starburst (high cognitive load from 22 to 8.5; low cognitive load from 7.5 to 3) and the range of angles used (high cognitive load from 137 to 72; low cognitive load from 55 to 25). This improvement in number of angle changes, possibly as a result of better knowing what range of angles to use, was correlated with the decrease in completion time from Starburst to Saw Blade, for both cognitive load conditions. Such results are consistent with a reduction of the number of cycles in the decision ladder for choosing angles.

Furthermore, there is some evidence of the low cognitive load condition participants better learning what angles to use. The results showed that the low cognitive load participants used a significantly smaller angle range (high: 137; low: 25) and significantly fewer angle adjustments (high: 8.5; Low: 3) in the Saw Blade problem than the high cognitive load participants did for Saw Blade. This result is evidence that the low cognitive load participants may have been better able to learn the range of useful angles while completing Starburst. Therefore, there is evidence that the participants in the low cognitive load condition learned how to better choose the angles needed.

The above discussion noted that the participants in both conditions learned to use Logo equally well, but with the participants in the low cognitive load learning more quickly for some problems. Furthermore, evidence has been provided that the participants in the study learned to identify the need for and to use subprograms and loops. Additionally, evidence of the participants learning of rules related to the adjustment of angles was discussed. This evidence supports the hypothesis that design of instructional material, based on CLT, can affect the rules that students learn and how well they learn them. After discussing the limitations of this study, the results of the study, as have just been discussed, will be summarized.

6.5 Limitations

This study has a number of limitations, including an error with the NASA-TLX measurements, limitations in the control of construct validity, issues related to the assessment and effects of experience, the design of the test problems, and the number of practice problems.

First, for this study, the NASA-TLX was administered using an iPad-based program. In the iPad implementation, the performance scale was anchored with the terms low and high, while in the original version (Hart and Staveland, 1988), the performance scale is anchored with the terms good and poor.

Second, as with the previous study, the effect of cognitive load has not been completely isolated. As noted in the beginning of this dissertation, this lack of isolation is common in human factors studies. As in the EID studies mentioned, i.e. Christoffersen, Hunter, & Vicente (1996) and Vicente, Christoffersen, & Pereklita (1995), the design of this study was based on the use of previously developed theory to apply certain interventions to a particular problem.

There are a number of examples where it might not be the effect of cognitive load that caused the difference in completion times. For instance, as mentioned, in the Bow Tie problem, the participants in the low cognitive load condition were told, as part of a completion problem, to complete the problem using subprograms. The effect of transfer shown in Bow Tie Two may then be due to either the adjustment of cognitive load or the wording of the problem, and the instructions to use subprograms; however, this distinction may be difficult to make in any completion problem. Similarly, in the H problem, the participants in the high cognitive load condition completed the problem faster and had a higher success rate. This difference could be due to the presentation of the problem which could have increased the difficultly of the problem.

A second limitation of the study relates to the division between experience levels of the participants. As was done in Clarke et al. (2005), the self-ratings of the participants were used to assess experience, and the participants were divided into two categories of experience level, with the division occurring at the middle of the scale. In both studies, the effect of condition was then measured for each experience level. Clarke et al. (2005) admitted, as I do, that a better assessment of previous knowledge could be useful; however, the current study does assess the relationship between self-assessment and other measures of experience, including number of programming courses taken, to establish validity for the self-assessment measure.

As a related issue, there was a small number of participants in the group with self-assessment > 1. The participants were separated to detect differences in the group with self-assessment ≤ 1 that might have been hidden by the participants with a higher level of experience. As a result of the small numbers, the tests for significant differences at self-assessment > 1 had low power, and were unlikely to detect a significant difference in completion times between conditions at the level of the sample differences. Therefore, conclusions about the usefulness of the instructional supports at the higher level of experience are difficult to make. On a related note, some of the statistical comparisons made in this study's analysis involved very small sample sizes. Such comparisons are useful as an exploratory tool; however, some future work should aim to increase sample sizes, potentially increasing the validity of comparisons made.

A third limitation relates to the similarity of instructional and testing problems. The study could have benefited from more questions in the testing section that were more similar to questions in the instructions. In such a case, there would be more opportunity for transfer of rules that could have been analysed, providing more insight into what types of rules the participants would have been capable of learning in a short period of time. Moreover, more similar problems might have resulted in a significant difference in completion time for the test sections of the study, because of greater opportunity for transfer difference between conditions.

Finally, the participants in this study could have benefited from additional practice over a broader range of examples, because as previously discussed, practice over a range of examples is needed for transfer. Relatedly, with additional practice problems that did not introduce new concepts, a wider differential of cognitive load between the two instructional conditions in the presentation of practice examples could have been used. In other words, more traditional problems could have been used in the high cognitive load condition relative to the low cognitive load condition. With such an increase in practice problems, with better differentiation between conditions, a greater effect of instructional condition may have been seen in the study, including in the tests.

6.6 Summary

This study was designed to test the transfer of knowledge learned from instructional materials based on a set of theoretically transferable rules derived from a WDA and CTA. During the study the participants completed a number of instructional, testing, and workload assessment steps. Cognitive load during instruction was reduced in one condition through worked examples and completion problems. The participants in the low cognitive load condition did not perform better on the tests as a result of the reduced cognitive load, but they did learn the material in less time, and in the case of the loop instructions with less workload, providing evidence that cognitive load controls can help with more efficient learning of the instructional material.

There was evidence of transfer between the instructions and the tests, and, in some cases, evidence of condition on what was transferred. For example, there was evidence of transfer pertaining to the recognition and use of subprograms and loops. Additionally, for the low cognitive load condition participants, there was evidence of better learning of knowledge related to the adjustment of angles in figures. This angle adjustment behaviour was theorized to be related to the rule-based navigation of a decision ladder.

This study filled in some of the gaps of the study in Chapter 5 by providing evidence of the transfer of knowledge to new problem-solving tasks. The evidence in the study supports the use of CLT-based interventions to promote more efficient learning, and in some cases, better transfer, of instructional materials based on the rules from a WDA and CTA analysis.

Chapter 7
Contributions and Conclusion

In this dissertation, I have examined the need to expand the use of learning science results into the design of training materials in a way that takes into account the cognitive components of a CWA analysis. CWA is an important tool for the analysis and design of complex systems, and training is an important component of CWA-based analysis (Naikar, 2009). In this dissertation, a review of CWA-related training literature was conducted, and existing training methods at the ecological end of CWA, related to the WDA and CTA phases, were noted. Authors such as Lintern and Naikar (1998), Naikar and Sanderson (1999), Naikar and Saunders (2003), Naikar (2006b, 2009), and McIlroy and Stanton (2012) have investigated training at the ecological end of CWA, particularly as related to the WDA and CTA phases. Those authors used WDA to examine what system constraints operators must learn about and used CTA to examine the causes of errors and methods of avoiding them. Despite the useful work pertaining to training based on the ecologically oriented phases of CWA, a gap was identified in training research related to the cognitively oriented phases of CWA, particularly WCA. Such a gap is important to bridge, because any CWA analysis has aspects related to cognitive as well as ecological concerns (Vicente, 1999).

In this chapter, the work in this dissertation that built towards and filled the gap in the CWA training literature will be reviewed. First, the existing training literature discussed in this dissertation will be summarized. Then the contribution of this dissertation towards bridging the CWA training gap will be described. Finally, limitations of this research and possible future work will be presented.

7.1 Reviewed Work

As was discussed in Chapter 1, the SRM Taxonomy is currently the main tool used in the WCA phase of CWA. Therefore, training research aimed at the cognitive end of CWA should focus on SRM-related training. In Chapter 2, a lack of training research related to training requirements and methods for SRM-based training was identified (Dankelman et al., 2004; Dankelman, 2008). In filling this gap, the work in this dissertation addressed both the identification of SRM-related training needs and methods of meeting those needs.

In preparation for developing methods of identifying SRM-related training needs, a review of literature related to SRM was conducted and revealed an important consideration about the meaning of expertise from an SRM perspective, in particular that expert behaviour when completing tasks can involve a combination of a number of different behaviour types (Rasmussen, 1986; Vicente, 1999). In using a combination of behaviours, an experienced operator should be able to use cognitively efficient rule- and skill- based behaviour when completing a familiar task but be able to fall back to the more cognitively demanding model-based behaviour when completing a novel task for which known skill- and rule- based behaviours will not work. However, as part of the model-based behaviour, an operator may still be able to apply rule-based behaviours for some subtasks.

In order to have the ability to combine behaviour types to achieve efficient, yet correct, behaviour in new situations, operators need to learn rules that will be applicable in new situations. The use of existing knowledge in new situations is the essence of training transfer, and, therefore, a review of training transfer literature was conducted. From the review, two main points for connecting the learning of rules to training transfer were noted: the building of a knowledge base of rules to be drawn on and the avoidance of applying rules in inappropriate contexts. From these connections of rule-based behaviours and training transfer literature, guidelines for identifying training needs were derived, a research contribution that is described in the next subsection.

In addition to identifying training needs, it was noted that training methods for meeting those needs are necessary. Therefore, existing work on training rule-based behaviours was reviewed, particularly the 4C/ID method (van Merriënboer, 1997) and CLT (e.g. Sweller et al., 2011). Some shortcomings of 4C/ID for some CWA-based applications were noted, particularly related to part-task training, but the usefulness of whole-task training, as advocated in 4C/ID, was noted. Whole-task training, provided through contextual examples, including contrasting examples, was identified as useful in promoting transfer (Bransford & Schwartz, 1999; Day & Goldstone, 2012). However, it was noted that whole-task training and contextual examples can cause high cognitive load and interfere with learning (Day & Goldstone, 2012). Therefore, methods based on CLT, such as worked examples, completion problems, variable-priority training (Siegel, 1989), and pre-training (Sweller et al. 2012), were identified as useful to reduce the cognitive load imposed by training materials.

The above reviewed work identified a gap in SRM-based training in the context of CWA and then considerations to observe when filling in this gap were noted. After the literature review, this dissertation then built on the reviewed work to provide training methods related to SRM, as will be described next.

7.2 Contributions

In order to expand training design for systems analysed and designed with CWA to the cognitive domain, the work in this dissertation contributes a framework for developing training material based on the SRM Taxonomy. The framework connects, in a novel way, results from a WDA and CTA to the identification of rule-based behaviours that are intended to be useful in future, unknown tasks. The framework also uses methods based on CLT to guide the development of training materials for the identified rules. Both aspects of the framework, the identification of rules and the development of training materials, are discussed below. Additionally, in providing examples for the use of the framework, a new use of CWA was demonstrated, namely the analysis of two mathematics-related programming environments.

7.2.1 Identifying Training Rules

The work in Chapter 4 in this dissertation focused on the identification of rules that could potentially be useful in future tasks. There were two novel aspects presented in Chapter 4 pertaining to the identification of such rules: general guidelines for identifying rules that may be transferable to future tasks and methods for determining rules for a specific system.

The guidelines for identifying rules were developed from the observation that when completing novel tasks, an operator engages in cognitively demanding model-based behaviour, but could benefit from being able to transfer less cognitively demanding rule-based behaviour to familiar subtasks. As mentioned above, the use of existing knowledge in new situations is a form of training transfer; therefore, the literature on training transfer was combined with the existing literature on SRM-based learning to develop three guidelines for transferable rules. The three guidelines pertained to the recognition of signs and mapping those signs to actions, the discernment of appropriate context for rules, and the chaining of rules and formation of generalizations.

The three guidelines discussed provide general guidance in the derivation of rules for training. In addition, a method of deriving rules for specific systems from a WDA and CTA was also demonstrated. The work in Chapter 4 demonstrated the use of the information from a WDA and CTA to develop a specific set of rules for the Maple computer algebra system. Another example of the derivation of rules from a WDA and CTA, for the programming language Logo, was given in Chapter

6. The guidelines used to guide the identification of training rules to encourage efficient rule-based behaviour constitutes a contribution to the CWA framework, allowing CWA to be used in a more complete way for the design of complex systems training. The examples of rule development contribute to the practice of CWA, showing how WDA and CTA can be used to extract rule-based behaviours for training.

7.2.2 Identifying Training Methods

Once the sets of rules were identified, training materials for the rules identified needed to be designed, and as noted, it is important to control the cognitive load in training material, leading to work in this dissertation concerning the application of CLT to the training of rules.

As discussed, the training of knowledge for transfer can benefit from the use of contextual examples (Day & Goldstone, 2012; Bransford & Schwartz, 1999), and in complex systems training it is helpful if the training examples constitute whole tasks (van Merriënboer, 1997; Wickens et al., 2012). However, such whole-task examples can impose too high of a cognitive load, impeding learning (van Merriënboer, 1997), so methods derived from CLT can be useful in promoting learning through reduced cognitive load (Sweller et al., 2011).

In this dissertation, CLT was integrated with CWA, as demonstrated through two examples of training materials that were developed from the rules described above and used two different types of CLT-based interventions. First, the rules for the use of the Maple system were developed into two sets of instructional materials, one of which was designed to impose a lower cognitive load by using a pre-training method (Clark et al., 2005; Sweller et al., 2011). In Chapter 5, it was demonstrated that the participants in the lower cognitive load condition better learned the rules relevant to using Maple. Second, the rules for use of the Logo language were also developed into two sets of instructional materials, one of which was again designed to impose a lower cognitive load through methods such as worked examples and completion problems (Sweller et al., 2011). In Chapter 6, it was demonstrated that participants who used the lower cognitive load instructions learned some of the instructional material more quickly and with less workload. Furthermore, evidence was seen for the transfer of some behaviours, such as the use of subprograms and loops, and, in some cases, a difference was seen in transfer caused by instructional condition. This result provides evidence that lowering cognitive load can improve the learning of material designed based on rule-based behaviours, an empirical contribution of this dissertation.

7.2.3 Expanding the Use of CWA

In this dissertation, CWA was applied to two domains related to the learning of mathematics and programming: CASs and Logo. In both these domains, the problems that students will eventually solve using the tools learned could potentially be different, to varying degrees, from the training problems used. Because CWA is designed to model work domains where the eventual use of the tools in the domain is unknown at design and training time, the use of CWA is suited to these domains. However, CWA has not been previously used for domains such as CASs and Logo; therefore, the application of CWA in this dissertation to these two domains is a novel expansion to the practice of CWA.

For each of these two domains, a WDA and CTA were conducted. In Chapter 3, a WDA and CTA for Maple were developed based on the results of the experience Maple user study and reasoning about the purpose of Maple. In Chapter 6, a WDA and CTA for Logo were developed by reasoning about the purpose and functioning of Logo.

The above two sets of WDAs and CTAs demonstrate the use of CWA in these domains; however, as will be discussed in Section 7.3, there is much more work to be done to develop more complete CWA-based models for domains such as CASs and Logo.

In summary, this dissertation has presented a new framework for the development of training materials that are focused on the cognitive needs of operators, as identified in a CWA and described using the SRM taxonomy. The framework includes methods for deriving training needs in terms of rules and methods for developing training materials from the rules identified. In particular, the overall pathway demonstrated, from WDA and CTA, to rule discovery, to training materials development, along with empirical evidence of transfer to novel problems, is a useful contribution with many implications for further research and development. In recognition that CWA is often used in complex environments, care was taken to pair the development of rules with realistic strategies for reducing cognitive load while training rule-based behaviour. Through these various steps, it was shown that CWA can be used to generate more efficient learning behaviour when used as part of an approach for developing training materials. Additionally, through the choice of work domains used in demonstrating the framework, the range of uses of CWA was expanded to include mathematical-related programming environments.

7.3 Limitations and Future Work

Limitations for each of the studies in this dissertation were discussed in the relevant chapters. In this section, some common limitations of the studies are drawn out, and possible future work to address those limitations is discussed. Additionally, limitations and future work pertaining to the theoretical work in this dissertation are also discussed.

7.3.1 Empirical Limitations and Future Work

There are four main limitations to the design of the empirical studies in this dissertation: limitations to the CWA models, experiment validity controls, the types of analyses used, and training time.

First, as discussed in the previous section, the CWA-based models developed in Chapters 3 and 6, for Maple and Logo respectively, represent new applications of CWA. However, the models presented are based on a limited analysis of the work domain, and could benefit from further analysis and validation. In particular, the examination of the models by experts, particularly for the Maple domain, could help to further develop and validate the models.

Second, as already mentioned, for the Maple instructional and Logo studies, there is not a tight control on the internal and construct validity of the interventions and measures. As discussed at the beginning of this dissertation, this lack of control is common in human factors studies, due to the goal of conducting studies that have higher ecological validity.

In order to resolve this first limitation, more studies that involve the methods in this dissertation being applied to other work domains could be conducted. For example, studies that involve EID (Vicente and Rasmussen, 1992; Vicente, 2002), a type of interface design that is a common use of SRM, could be conducted. The EID application area could benefit from the methods in this dissertation, because EID-based interface design involves a number of aspects that would lend themselves well to the framework for learning needs determination described in this dissertation.

Vicente and Rasmussen (1992) noted that EID displays are designed to support several types of behaviours, including rule- and model- based behaviours, as they are needed; however, Vicente (1992) pointed out that a goal of EID design is to encourage more rule-based behaviours when possible. The authors noted that, in order to support rule-based behaviours when they are applicable, EID interfaces are designed to provide sign-based cues to the operators. Therefore, the first guideline of design presented in Chapter 4, related to sign recognition and action mapping, would connect well with developing training materials for this aspect of EID interfaces. However, Vicente and

Rasmussen (1992) also claimed that there is a potential danger in operators using inappropriate rule-based behaviours with an EID interface, so, as identified as well in Chapter 4, training related to EID interfaces should involve the determination of appropriate contexts and constraints for rules.

There is also a connection of EID to the abstraction hierarchy in a method similar to the connection of the work in this dissertation to the abstraction hierarchy. To assist operators in avoiding the misapplication of rules in the operation of an EID interface, Vicente and Rasmussen (1992) proposed the use of the WDA abstraction hierarchy in developing interface features to help operators be cognizant of the constraints in the system that must not be violated. Similarly, in this dissertation, to fill the requirements of the second guideline from Chapter 4, the use of the WDA in developing training needs relevant to the constraints of rules was proposed. Therefore, the methods of identifying constraints in developing training materials could be applied to the training of operators in the observance of EID interface constraints.

Because the derivation of rules in the work in this dissertation is based on the abstraction hierarchy, and the guidelines in Chapter 4 have parallels in EID design, the methods in this dissertation for training rule derivation are possibly applicable to determining training needs for EID. Furthermore, the additional research and results, by testing the methods proposed in this dissertation on the design of training for EID displays, could help bolster the results in this dissertation by providing convergent evidence.

A third limitation of the empirical work pertains to methods of analysis used in this dissertation. In particular, the cognitive processes used by the participants to complete the tasks in the studies were not discussed; therefore, the exact rules used by participants during the studies are unknown. It was previously discussed that the analysis of verbalizations can be difficult because operators often do not have explicit, rational reasons for rule-based behaviours (Rasmussen, 1979). However, Rasmussen (1976) did explore some verbalization, and such a future analysis could provide some evidence that operators explicitly learned and used the rules developed with the methods in this dissertation.

As a fourth limitation, the participants in the studies in this dissertation could have benefited from more training time. As Haskell (2001) noted, people can spend hundreds, or even thousands, of hours practicing skills to become experts in a domain (see also Ericsson, Krampe, & Tesch-Römer, 1993). That fact does not necessarily mean that hundreds of hours of practice are needed to see an effect of a piece of instructional material, but more practice than the participants had in the studies in this dissertation could be useful. For example, for the Logo study, it was noted that more training tasks, which were differentiated between the conditions based on instructional interventions, could have resulted in a greater difference in test performance between the conditions. Therefore, additional research into longer-term training programs design using the methods presented in this dissertation may be useful.

The above limitations and the proposed resolutions require additional empirical research.   Such research would require the development of further training rules and then training materials, and such development could benefit from some additional theoretical work into training material design from a set of rules, as discussed next.

7.3.2 Future Theoretical Work

From a theoretical perspective, additional research could focus on transforming training rules into instructional material. Such a further development could follow a similar path as the work on EID. As Vicente (2002) pointed out in the context of EID, the design of interfaces from the results of a CWA is still largely a creative effort, though work such as Burns and Hajdukiewicz (2004) has helped to fill this gap and add some regularity to the design of EID interfaces. Likewise, the design of instructional material from a CWA is still very much a creative endeavour that involves the mixing of information from the CWA and domain-specific knowledge; however, further work may help make the design of instructional material a more regular and defined process.

The output from a design method could also be affected by the choice of instructional technique. In this dissertation, the instructional materials used in the studies were based on examples. The participants in the studies had to draw any generalizations from the examples, a process that can be difficult and cognitively demanding (van Merriënboer, 1997). Instead, using instructional methods that make generalizations more explicit may help operators transfer rules to new situations (see for example, Day and Goldstone, 2012; Bransford & Schwartz, 1999). Therefore, additional theoretical work into the design of other types of instructional materials from the sets of transferable rules could be beneficial.

There is also a potential for the theoretical work into training design for CWA to feed back into the learning science and design literature. In Chapter 3, the limitations of the 4C/ID method as it pertained to the design of instructional materials for teaching rules was discussed; however, the use of whole-task training and instructional support was drawn out of 4C/ID for use in the design of training materials for CWA. For systems where 4C/ID is appropriate, the work in this dissertation may have a contribution to make to 4C/ID-based analysis. This dissertation uses the SRM inventory (Kilgore & St-Cyr, 2006) to identify rules, a method not seen in the 4C/ID literature. Such explicitly identified rules are key components in system analysis that uses the SRM taxonomy, such as CWA (Rasmussen, 1986; Kilgore & St-Cyr, 2006). A focus on explicitly identified SRM-based behaviours could complement the 4C/ID method, by giving a new method of training needs analysis, and the results of such an analysis may be useable in 4C/ID-based designs.

This section has presented a number of empirical and theoretical limitations that lead to future work. However, the two categories are not independent, because additional theoretical work into methods of designing new instructional materials from a set of rules could lead to additional empirical work, which could, in turn, provide more evidence for the usefulness of the designs created using the methods proposed in this dissertation.

7.3.3 Conclusion

This dissertation identified a gap in the training literature related to cognitive components of CWA. In particular, if CWA is an effective approach for understanding cognitive work, it should be possible to develop training materials from CWA that result in more efficient operator behaviour in novel situations. To begin to fill this gap, I proposed three guidelines for deriving transferable training rules that theoretically could be used in future scenarios as part of model-based behaviour. I also provided two examples of the use of these guidelines and used them in the design of instructional materials, which were developed using CLT-based design guidelines. The examples of instructional design also demonstrated the connection of the three guidelines to the WDA and CTA phases of CWA. Finally, two studies showed evidence of the usefulness of the methods presented in this dissertation in the development of a set of training rules and of CLT-based instructional methods to develop those rules into instructional materials.

Further examples of the use of these guidelines in other work domains and testing of the development of training materials could provide further evidence to the usefulness of the given guidelines in guiding rule derivation from a CWA and subsequent instructional design. Future work also remains to be done in developing methods of designing instructional material based on the guidelines and resulting sets of rules. With further study, the use of the guidelines and methods presented in this dissertation should help improve the design of training materials for the complex systems commonly analysed with CWA.

Bibliography

Abelson, H. (1981). *Turtle Geometry.* Cambridge, MA: MIT Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036-60.

Artigue, L. E. (2002). Learning mathematics in a CAS environment: the genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning*, 7, 245-274.

Ashby, W.R. (1958). Requisite variety and its implications for the control of complex systems. *Cybernetica* 1(2), 83-99.

Ashoori, M., & Burns, C. (2010). Reinventing the Wheel: Control Task Analysis for Collaboration. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *54*(4), 274–278. doi:10.1177/154193121005400402

Ashoori, M., Burns, C., Momtahan, K., & d' Entremont, B. (2011). Control Task Analysis in Action: Collaboration in the Operating Room. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *55*(1), 272–276.

Ashoori, M., & Burns, C. (in press). Team Cognitive Work Analysis : Structure and Control Tasks. *Journal of Cognitive Engineering and Decision Making*.

Ashoori, M. (2012). Cognitive Work Analysis to Support Collaboration in Teamwork Environments. (Unpublished Doctoral Thesis). University of Waterloo, Waterloo, Canada.

Ayres, P. (2006a). Impact of reducing intrinsic cognitive load on learning in a mathematical domain. *Applied Cognitive Psychology*, *20*(3), 287–298.

Ayres, P. (2006b). Using subjective measures to detect variations of intrinsic cognitive load within problems. *Learning and Instruction*, *16*(5), 389–400.

Bartlett, F.C. (1932). Remembering A Study in Experimental and Social Psychology. Cambridge, UK: University Press.

Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies. 14*, 237–249.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking Transfer: A Simple Proposal with Multiple Implications. *Review of Research in Education*, *24*, 61-10.

Brusilovsky, P., Calabrese, E., Kouchnirenko, A., & Miller, P. (1997). Mini-languages : a way to learn programming principles. *Education and Information Technology*, *2*, 65–83.

Burns, C.M. & Bisantz, A.M. (2008). Advances in the Application of Cognitive Work Analysis. InBisantz, A. and Burns, C.M. (Eds.), *Applications of Cognitive Work Analysis*. (pp. 3-18). Mahwah, NJ: Lawrence Erlbaum and Associates.

Burns, C.M., Hajdukiewicz, J.R. (2004). *Ecological Interface Design*. Boca Raton, FL: CRC Press.

Burns, C. M., Skraaning, G., Jamieson, G. A., Lau, N., Kwok, J., Welch, R., & Andresen, G. (2008). Evaluation of Ecological Interface Design for Nuclear Process Control: Situation Awareness Effects. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *50*(4), 663–679.

Buteau, C., Marshall, N., Jarvis, D. H, & Lavicza, Z. (2010). Integrating Computer Algebra Systems in post-secondary mathematics education: Preliminary results of a literature review. *International Journal for Technology in Mathematics Education, 17*(2), 57-68.

Chandler, P., & Sweller, J. (1991). Cognitive Load Theory and the Format of Instruction. *Cognition and Instruction*, 8(4), 293-332.

Chandler, P., & Sweller, J. (1996). Cognitive Load While Learning to Use a Computer Program. *Applied Cognitive Psychology*, *10*, 151–170.

Chase, G., & Simon, H. A. (1973). Perception in Chess. *Cognitive Psychology*, *4*, 55–61.

Chen, M. (1995). A methodology for characterizing computer-based learning environments. *Instructional Science*, *23*(1-3), 183–220. doi:10.1007/BF00890451

Christoffersen, K., Hunter, C. N., & Vicente, K. J. (1996). A Longitudinal Study of the Effects of Ecological Interface Design on Skill Acquisition. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *38*(3), 523–541.

Clarke, T., Ayres, P., & Sweller, J. (2005). The impact of sequencing and prior knowledge on learning mathematics through spreadsheet applications. *Educational Technology, Research and Development*, 53(3), 15–24.

Collins, A., Brown, J., & Newman, S. (1987). *Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics* (pp. 1–42). Retrieved from: http://ocw.metu.edu.tr/pluginfile.php/9107/mod_resource/content/1/Collins%20report.pdf

Collins, A. (2006). Cognitive Apprenticeship. In R. K. Sawyer (Ed.), *Handbook of the Learning Science* (pp. 47–60).

Cook, D. A, Beckman, T. J., Thomas, K. G., & Thompson, W. G. (2008). Introducing resident doctors to complexity in ambulatory medicine. *Medical education*, *42*(8), 838–48.

Cooper, A., Reimann, R., Cronin, D. (2007). *About Face 3.* New York, NY: Wiley.

Cornelissen, M., Salmon, P. M., Jenkins, D. P., & Lenné, M. G. (2012, in press). A structured approach to the strategies analysis phase of cognitive work analysis. *Theoretical Issues in Ergonomics Science*.

Cornelissen, M., Salmon, P. M., Jenkins, D. P., & Lenne, M. G. (2011). How can they do it? A structured approach to the strategies analysis phase of Cognitive Work Analysis. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *55*, 340–344.

Cornelissen, M., Salmon, P. M., & Young, K. L. (in press, 2012). Same but different? Understanding road user behaviour at intersections using cognitive work analysis. *Theoretical Issues in Ergonomics Science*.

Cowan, N. (2001). The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24, 87-114.

Crandall, B., Klein, G., & Hoffman, R., (2006). *Working Minds: A Practitioner's Guide to Cognitive Task Analysis.* Cambridge, MA: The MIT Press.

Creswell, J.W. & Clark, V.L. (2007). *Mixed Methods Research.* Thousands Oaks, CA: Sage Publications.

Dainoff, M. J., Mark, L. S., Hall, C., & Richardson, A.R. (2002). Cognitive Work Analysis in Education and Training: Relating Pedagogical Methods to Course Objectives. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *46*, 825–828.

Dankelman, J., Wentink, M., Grimbergen, C. a, Stassen, H. G., & Reekers, J. (2004). Does virtual reality training make sense in interventional radiology? Training skill-, rule- and knowledge-based behavior. *Cardiovascular and Interventional Radiology*, *27*(5), 417-21.

Dankelman, J. (2008). Surgical simulator design and development. *World Journal of Surgery*, *32*(2), 149–55.

Darken, R. (2009). Identifying the limits of training system effectiveness through taxonomies of human performance. *Theoretical Issues in Ergonomics Science*, *10*(3), 231–243.

De Groot, A.D. (1965). Thought and Choice in Chess. New York, NY: Basic Books, Inc.

DeLeeuw, K. E., & Mayer, R. E. (2008). A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load. *Journal of Educational Psychology*, *100*(1), 223–234.

Detterman, D.K. (1993). The case for the prosecution: Transfer as an epiphenomenon. In D.K. Detterman, R.J. Sternberg (Eds.), *Transfer on Trial: Intelligence, Cognition, and Instruction* (pp. 1-24). Westport, CT: Ablex Publishing.

Drivalou, S., & Marmaras, N. (2009). Supporting skill-, rule-, and knowledge-based behaviour through an ecological interface: An industry-scale application. *International Journal of Industrial Ergonomics*, *39*(6), 947–965.

Dreyfus, S., & Dreyfus, H. (1980). A five-stage model of the mental activities involved in directed skill acquisition. *Washington, DC: Storming Media*. Retrieved June 13, 2010.

Dreyfus, H.L., & Dreyfus, S.E. (1988). *Mind over Machine: The power of human intuition and expertise in the era of the computer.* New York: The Free Press.

Elix, B., & Naikar, N. (2008). Designing safe and effective future systems: A new approach for modelling decisions in future systems with Cognitive Work Analysis. *Proceedings of the 8th International Symposium of the Australian Aviation Psychology Association.* Sydney, Australia: Australian Aviation Psychology Association.

Endsley M.R., Bolté B., & Jones D.G. (2003). *Designing for Situation Awareness: An Approach to User-Centered Design.* Boca Raton, FL: CRC Press.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological review*, *100*(3), 363–406.

Feurzeig, W., Papert, S.A. & Lawler, B. (2011 reprint of 1968 paper). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments.* 19(5), 487-501.

Feurzeig, W. (2010). Toward a Culture of Creativity: A Personal Perspective on Logo's Early Years and Ongoing Potential. *International Journal of Computers for Mathematical Learning*, *15*(3), 257-265.

Fowlkes, J. E., Neville, K. J., Owens, J. M., & Hafich, A. J. (2009). Challenges to the development of pedagogically driven engineering requirements for complex training systems. *Theoretical Issues in Ergonomics Science*, *10*(3), 217–229.

Gerjets, P., Scheiter, K., & Catrambone, R. (2004). Designing Instructional Examples to Reduce Intrinsic Cognitive Load: Molar versus Modular Presentation of Solution Procedures. *Instructional Science*, *32*(1/2), 33–58.

Gerjets, P., Scheiter, K., & Catrambone, R. (2006). Can learning from molar and modular worked examples be enhanced by providing instructional explanations and prompting self-explanations? *Learning and Instruction*, *16*(2), 104–121.

Gibson, E.J., Pick, A.D. (2000). *An Ecological Approach to Perceptual Learning and Development.* New York, NY: Oxford University Press.

Gibson, J.J. (1977). The theory of affordances. In R. Shaw & J. Bransford (Eds.), *Perceiving, acting, and knowing: Toward an ecological psychology* (pp. 67-82). Hillsdale, NJ: Erlbaum.

Gibson, J. J. (1979). *The Ecological Approach to Approach to Visual Perception. Perceiving, Acting and Knowing*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Gopher, D., Weil, M., & Siegel, D. (1989). Practice under changing priorities: An approach to the training of complex skills. *Acta Psychologica*, *71*(1-3), 147–177.

Hager, P., & Hodkinson, P. (2009). Moving beyond the metaphor of transfer of learning. *British Educational Research Journal*, *35*(4), 619-638.

Hart, S. G. (2006). Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *50*(9), 904–908.

Hart, S.G., & Staveland, L.E. (1988). Development of the NASA-TLX: Results of Empirical and Theoretical Research. In P.A. Hancock, N. Meshkati (Eds.), *Human Mental Workload.* (pp. 139-184). The Netherlands: Elsevier Science Publishers.

Harvey, B. (1997). *Computer Science Logo Style. Volume 1-3*. Cambridge, Mass.; London: MIT.

Haskell, R.E. (2001). *Transfer of Learning*. London: Academic Press.

Hollnagel, E., & Woods, D. (1983). Cognitive systems engineering: New wine in new bottles. *International Journal of Man-Machine Studies*, *51*, 339–356.

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction*, *7*(2), 174–196.

Hajdukiewicz, J. R., Vicente, K. J., Doyle, D. J., Milgram, P., & Burns, C. M. (2001). Modelling a medical environment: an ontology for integrated medical informatics design. *International journal of medical informatics*, *62*(1), 79–99.

Hajdukiewicz, J., & Vicente, K. (2004). A theoretical note on the relationship between work domain analysis and task analysis. *Theoretical Issues in Ergonomics. 5*(6), 527-538.

Hobbs, A., & Williamson, A. (2002). Skills, rules and knowledge in aircraft maintenance : errors in context. *Ergonomics*, *45*(4), 290–308.

Hockey, G. R. J., Sauer, J., & Wastell, D. G. (2007). Adaptability of Training in Simulated Process Control: Knowledge- Versus Rule-Based Guidance Under Task Changes and Environmental Stress. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *49*(1), 158–174.

Hutchins, E. (1995). *Cognition in the Wild*. Cambridge, MA: The MIT Press.

International Ergonomics Association (2010). http://www.iea.cc/

Järvelä, S. (1995). The Cognitive Apprenticeship Model In A Technologically Rich Learning Environment : Interpreting The Learning Interaction. *Learning and Instruction*, *5*, 327–259.

Jenkins, D. P., Stanton, N. A., Salmon, P. M., Walker, G. H., & Young, M. S. (2008). Using cognitive work analysis to explore activity allocation within military domains. *Ergonomics*, *51*(6), 798–815.

Jenkins, D., Stanton, N., Walker, G., Salmon, P., & Young, M. (2010). Using cognitive work analysis to explore system flexibility. *Theoretical Issues in Ergonomics Science*, *11*(3), 136–150.

Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The Expertise Reversal Effect. *Educational Psychologist*, 38, 37-41.

Kehoe, J.E., Bednall, T. C., Yin, L., Olsen, K. N., Pitts, C., Henry, J. D., & Bailey, P. E. (2009). Training adult novices to use computers: Effects of different types of illustrations. *Computers in Human Behavior*, *25*(2), 275–283.

Kilgore, R., & St-Cyr, O. (2006). The SRK Inventory: A Tool for Structuring and Capturing a Worker Competencies Analysis. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *50*, 506–509.

Kilgore, R., St-Cyr, O., and Jamieson, G.A. (2009). From work domains to worker competencies: a five-phase CWA. In: A.M. Bisantz and C.M. Burns, eds. *Applications of cognitive work analysis*. Boca Raton, FL: CRC Press, 15–47.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work : An Analysis of the Failure of Constructivist, Based Teaching. *Educational Psychologist*, 41(2), 75-86.

Klein, G., Calderwood, R., & MacGregor, D. (1989). Critical Decision Method for Eliciting Knowledge. *IEEE Transactions on Systems, Man, and Cybernetics*, *19*(3), 462–472

Klein, G. (1999). *Sources of Power: How People Make Decisions.* Cambridge, MA: MIT Press.

Kilpatrick, J., Swafford, J., & Findell, B. (2001). *Adding it up: Helping Children Learn Mathematics*. Washington, DC: National Academy Press.

Kim, S. K., Suh, S. M., Jang, G. S., Hong, S. K., & Park, J. C. (2012). Empirical research on an ecological interface design for improving situation awareness of operators in an advanced control room. *Nuclear Engineering and Design*, *253*, 226–237.

Lamoureux, T.M. & Chalmers, B.A. (2009). Control Task Analysis: Methodologies for Eliciting and Applying Decision Ladder Models for Command and Control. In A.M. Bisantz, C.M. Burns (Eds.) *Applications of Cognitive Work Analysis* (pp. 95-128). Boca Raton: CRC Press.

Lavicza, Z. (2007). Factors influencing the integration of Computer Algebra Systems into university-level mathematics education. *International Journal for Technology in Mathematics Education*. 14(3) 121-129.

Lavicza, Z. (2010). Integrating technology into mathematics teaching at the university level. *Zdm*, *42*(1), 105-119.

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

Lin, C. J., Yenn, T.C., & Yang, C.W. (2010). Optimizing human–system interface automation design based on a skill-rule-knowledge framework. *Nuclear Engineering and Design*, *240*(7), 1897-1905.

Lintern, G., Naikar, N. (1998). Cognitive Work Analysis for Training System Design. *Computer Human Interaction Conference Proceedings*, 252–259.

Lintern, G. (2000). An Affordance-Based Perspective on Human-Machine Interface Design An Affordance-Based Perspective on Human – Machine Interface Design. *Ecological Psychology*, *12*(1), 65–69.

Lintern, G., Naikar, N. (2000). The use of work domain analysis for the design of training systems. *Proceedings of the HFES/IEA 2000,* 198–201.

Lintern, G. (2009). The Theoretical Foundation of Cognitive Work Analysis. In A.M. Bisantz, C.M. Burns (Eds.) *Applications of Cognitive Work Analysis* (pp. 321-355). Boca Raton: CRC Press.

Lintern, G. (2010). A Comparison of the Decision Ladder and the Recognition-Primed Decision Model. *Human Factors*, *4*(4), 304–327.

Long, J. (2000). Cognitive Ergonomics - Past, Present, Future: 10 Lessons Learned (10 Lessons Remaining). *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *44*(6), 557–560.

Marmaras, N., & Nathanael, D. (2005). Cognitive engineering practice: melting theory into reality. *Theoretical Issues in Ergonomics Science*, *6*(2), 109–127.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *The American Psychologist*, *59*(1), 14-9.

Mayer, R.E. (2003). Mathematical Problem Solving. In J.M. Royer (eds.) *Mathematical Cognition.* (pp. 69-92). Greenwich, CT: Information Age Publishing.

Mayer, R.E. (2005). Conceptual Understanding Versus Computational Skill. In J.M. Royer (Ed.) *The Cognitive Revolution in Educational Psychology* (pp. 105-119). Greenwich, CN: Information Age Publishing.

McIlroy, R. C., & Stanton, N. A. (2011). Getting past first base: Going all the way with Cognitive Work Analysis. *Applied ergonomics*, *42*(2), 358–70.

Mcilroy, R. C., & Stanton, N. A. (2012). Specifying the requirements for requirements specification : the case for Work Domain and Worker Competencies Analyses. *Theoretical Issues in Ergonomics Science*, *13*(4), 450–471.

Merlino, N., & Rhodes, R. (2012). Technology in the 21st Century Classroom: Key Pedagogical Strategies for Millennial Students in University Business Courses. *Journal of Supply Chain and Operations*, *10*(1), 113-130.

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.

Mumaw, R., Roth, E., Vicente, K., & Burns, C. (2000). There is More to Monitoring a Nuclear Power Plant than Meets the Eye, *Human Factors*, 42(1), 36-55.

Naikar, N. (2006a). An Examination of the Key Concepts of the Five Phases of Cognitive Work Analysis with Examples from a Familiar System. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *50*(3), 447–451.

Naikar, N. (2006b). Beyond interface design: Further applications of cognitive work analysis. *International Journal of Industrial Ergonomics*, *36*(5), 423–438.

Naikar, N. (2009). Beyond the Design of Ecological Interfaces: Applications of Work Domain Analysis and Control Task Analysis to the Evaluation of Design Proposals, Team Design, and Training. In A.M. Bisantz, C.M. Burns (Eds.) *Applications of Cognitive Work Analysis* (pp. 69-94). Boca Raton: CRC Press.

Naikar, N. (2011). Cognitive Work Analysis: Foundations, Extensions, and Challenges. Australian Department of Defense. Retrieved from: http://www.ntis.gov/search/product.aspx?ABBR=ADA564221

Naikar, N., Moylan, A., & Pearce, B. (2006). Analysing activity in complex systems with cognitive work analysis: concepts, guidelines and case study for control task analysis. *Theoretical Issues in Ergonomics Science*, *7*(4), 371–394.

Naikar, N., Pearce, B., Drumm, D., & Sanderson, P. M. (2003). Designing Teams for First-of-a-Kind, Complex Systems Using the Initial Phases of Cognitive Work Analysis: Case Study. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *45*(2), 202–217.

Naikar, N., & Sanderson, P. M. (1999). Work domain analysis for training-system definition and acquisition. *The International Journal of Aviation Psychology*, *9*(3), 271–290.

Naikar, N., & Saunders, A. (2003). Crossing the boundaries of safe operation: An approach for training technical skills in error management. *Cognition, Technology & Work*, *5*(3), 171–180.

NASA (n.d.). NASA TLX Paper/Pencil Version. *NASA TLX: Task Load Index.* Retrieved: August 13, 2012. http://humansystems.arc.nasa.gov/groups/TLX/paperpencil.html

Newell, A. & Simon, H.A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice Hall.

Norman, D.A. (1988). The Psychology of Everyday Things. New York, NY: Basic Books.

Olsen, S.E., Rasmussen, J. (1989). The Reflective Expert and the Prenovice: Notes on Skill-, Rule-, and Knowledge- based Performance in the Setting of Instruction and Training. In: L. Bainbridge and S. A.R. Quintanilla (eds.) *Developing Skills with Information Technology* (pp. 9-33). New York, NY: John Wiley & Sons.

Paas, F., & Gog, T. V. (2009). Principles for Designing Effective and Efficient Training of Complex Cognitive Skills. *Human Factors*, 5, 166–194.

Papert, S. (1980). Mindstorms: children, computers, and powerful ideas. New York, NY: Basic Books, Inc.

Plato (1977 trans.). The Republic. (B. Jowett trans.) New York, NY: Penguin Press.

Pérez-Sanagustín, M., Santos, P., Hernández-Leo, D., & Blat, J. (2012). 4SPPIces: A case study of factors in a scripted collaborative-learning blended course across spatial locations. *International Journal of Computer-Supported Collaborative Learning*, *7*(3), 443–465.

Pierce, R., & Stacey, K. (2001). Observations on Students' Responses to Learning in a CAS Environment Important Features of Learning With CAS Gain Broad Experience : Use Multiple Representations and Many Examples. *Mathematics Education Research Journal*, *13*(1), 28-46.

Pierce, R., Herbert, S., & Giri. J. (2004). CAS: Student engagement requires unambiguous advantages. *Proc. of the 27th Annual Conf. of the Mathematics Education Research Group of Australasia,* 462–469.

Pohl, M., Smuc, M., & Mayr, E. (2012). The User Puzzle—Explaining the Interaction with Visual Analytics Systems. *IEEE Transactions on Visualization and Computer and Computer Graphcis*, *18*(12), 2908-2916.

Proctor, R.W., Van Zandt, T. (2008). *Human Factors in Simple and Complex Systems.* Boca Raton, FL: CRC Press.

Renkl, A., Freiburg, D., Atkinson, R. K., & Maier, U. H. (2000). From Studying Examples to Solving Problems: Fading Worked-Out Solution Steps Helps Learning How to Combine Example Study and Problem. *Proceedings of the 22nd Annual Conference of the Cognitive Science Society,* 393-398.

Reason, J. (1990). Human Error. Cambridge: Cambridge University Press.

Rasmussen, J. (1976). Outlines of a hybrid model of the process plant operator. In T.B. Sheridan and G. Johannsen (Eds.), *Monitoring Behavior and Supervisory Control*, (pp. 371–384). New York: Plenum Press.

Rasmussen, J. (1983). Skills, Rules, and Knowledge ; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models. *IEEE Transactions on Systems, Man, and Cybernetics.* (3), 257-266.

Rasmussen, J. (1985). The role of hierarchical knowledge representation in decision making and system management. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(2), 234–243.

Rasmussen, J. (1986). *Information processing and human-machine interaction.* New York, NY: Elsevier Science Pub. Co. Inc.

Rasmussen, J., Pejtersen, A.M., Goodstein, L.P. (1994). *Cognitive Systems Engineering.* New York, NY: Wiley.

Rasmussen, J., & Vicente, K. J. (1989). Coping with human errors through system design: implications for ecological interface design. *International Journal of Man-Machine Studies*, *31*(5), 517–534.

Reed, S.K. (1993). A Schema-based Theory of Transfer. In D.K. Detterman, R.J. Sternberg (Eds.), *Transfer on Trial: Intelligence, Cognition, and Instruction* (pp. 39-67). Westport, CT, US: Ablex Publishing.

Robinson, T., Burns C. (2009). Computer Algebra Systems and Their Effect on Cognitive Load. *Proceedings of the 9th bi-annual international conference on naturalistic decision making NDM9: NDM and Computers.* London, UK. June 23-26, 2009.

Robinson, T., Burns, C.M. (2010). A Learning Process: Old Strategies, New Tools. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* 54, 384-388.

Royer, J.M., Mestre, J.P., Dufrense, R.J (2005). Framing the transfer problem. In J. P. Mestre (Ed.) *Transfer Of Learning From A Modern Multidisciplinary Perspective* (pp. vii-xvii). Greenwich, Ct: Information Age.

Salmon, P., & Jenkins, D. (2010). Hierarchical task analysis vs. cognitive work analysis : comparison of theory, methodology and contribution to system design. *Theoretical Issues in Ergonomic Science*, 11(6), 37–41.

Sanderson, P.M., Harwood, K. (1988). The skills, rules, and knowledge classification: a discussion of its emergence and nature. In L.P. Goodstein, H.B. Anderson, S.E. Olson (Eds.) *Tasks, Errors, and Mental Models* (pp. 21-34). London: Taylor & Francis.

Schnotz, W., & Kürschner, C. (2007). A Reconsideration of Cognitive Load Theory. *Educational Psychology Review*, 19(4), 469-508.

Schwartz, D. L., Bransford, J. D., & Sears, D. (2005). Efficiency and Innovation In Transfer. In J. P. Mestre (Ed.) *Transfer Of Learning From A Modern Multidisciplinary Perspective* (pp. 1-51). Greenwich, Ct: Information Age.

Serway, R.A. (1996). *Physics for Scientists and Engineers 4th Edition.* Philadelphia, PA: Saunders College Publishing.

Sharp, T. D., & Helmicki, A. J. (1998). The Application of the Ecological Interface Design Approach to Neonatal Intensive Care Medicine. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *42*(3), 350–354.

Schneider, W., & Shiffrin, R. M (1977). Controlled and Automatic Human Information Processing: I. Detection, Search, and Attention. *Psychological Review*, *84*(1), 1-66.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and Automatic Human Information Processing: II. Perceptual Learning, Automatic Attending, and a General Theory. *Psychological Review*, *84*(2), 127-190.

Simon, D., & Simon, H. (1978). Individual differences in solving physics problems. In R. Siegler, (Ed.), *Children's thinking: What develops?* (pp. 40-74). NJ: Erlbaum.

Singley, M.K., & Anderson, J.R. (1989). *The Transfer of Cognitive Skill*. Cambridge, MA: Harvard University Press.

Stewart, J. (1995). Calculus 3rd Edition. Pacific Grove, CA: Brooks/Cole Publishing Company.

St.Pierre, M., Hofinger, G., Simon, R., & Buerschaper, C. (2008). *Crisis Management in Acute Care Settings*. Berlin: Springer.

Sweller, J., & Cooper, G. A. (1985). The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. *Cognition and Instruction*, *2*(1), 59-89.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257-285.

Sweller, J., & Chandler, P. (1991). Evidence for Cognitive Load Theory. *Cognition and Instruction*, 8(4), 251-362.

Sweller, J., & Chandler, P. (1994). Why Some Material Is Difficult to Learn. *Cognition and Instruction*, *12*(3), 185-233.

Sweller, J. (1994). Cognitive Load Theory, learning difficulty, and instructional design. *Learning and Instruction,* 4, 295-312.

Sweller, J. (2010). Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load. *Educational Psychology Review*, 22 (2), 123-138.

Sweller, J., Ayres, P., and Kalyuga, S. (2011). *Cognitive Load Theory*. New York, NY: Springer.

Tarmizi, R. a., & Sweller, J. (1988). Guidance during mathematical problem solving. *Journal of Educational Psychology*, *80*(4), 424-436.

Thorndike, E. (1906). *Principles of Teaching*. Syracuse, NY: The Mason-Henry Press.

Thorndyke, P.W. (1984). Applications of Schema Theory in Cognitive Research. In J.R. Anderson and S.M. Kosslyn (Eds.) *Tutorials in Learning and Memory* (pp. 167-191). San Francisco, CA: W.H. Freeman and Company.

Tjiam, I. M., Schout, B. M. A, Hendrikx, A. J. M., Scherpbier, A. J. J. M., Witjes, J. A., & Van Merriënboer, J. J. G. (2012). Designing simulator-based training: An approach integrating cognitive task analysis and four-component instructional design. *Medical teacher*, *34*(10), 698–707.

Torenvlient, G. (2003). We Can't Afford It. *Interactions*, *10*(4), 12–17.

Trochim, W.M.K., Donnelly, J.P. (2008). *The Research Methods Knowledge Base.* Mason, OH: Atomic Dog.

Univeristy of Waterloo (2009). *Math 137 Curriculum.* Retrieved 23 January, 2013, from: http://csclub.uwaterloo.ca/~mjessome/courses/1099/math137.pdf

van Merriënboer, J.G. (1990). Strategies for Programming instruction in High School: Program Completion vs. Program Generation. *Educational Computing Research.* 6(3), 265-286.

van Merriëboer, J.J.G. (1997). *Training Complex Cognitive Skills: A Four-Component Instructional Design Model For Technical Training*. Englewood Cliffs, NJ: Educational Technology Publications Inc.

van Merriënboer, J. J. G., Jelsma, O., & Paas, F. G. W. C. (1992). Training for reflective expertise: A four-component instructional design model for complex cognitive skills. *Educational Technology Research and Development*, *40*(2), 23–43.

van Merrienboer, J.J.G., Kirschner, P.A. & Kester, L. (2003). Taking the Load Off a Learner's Mind: Instructional Design for Complex Learning. *Educational Psychologist*. 38(1), 5-13.

van Merriënboer, J.G., & Krammer, H. P. M. (1987). Instructional Strategies and Tactics for the Design of Introductory Computer Programming Courses in High School. *Instructional Science*, *16*, 251-285.

van Merriënboer, J. J. G., Clark, R. E., & Croock, M. B. M. (2002). Blueprints for complex learning: The 4C/ID-model. *Educational Technology Research and Development*, *50*(2), 39-61.

van Merrienboer, J. J. G., Kirschner, P. A., & Kester, L. (2003). Taking the Load Off a Learner's Mind: Instructional Design for Complex Learning. *Educational Psychologist*, *38*(1), 5–13.

Vicente, K.J. (1999). Cognitive Work Analysis: Towards safe, productive, and healthy computer-based work. Mahwah, NJ: Lawrence Erlbaum Associates.

Vicente, K. J. (2002). Ecological Interface Design: Progress and challenges. *Human Factors*, 44, 62-78.

Vicente, K. J., Christoffersen, K., & Pereklita, A. (1995). Supporting operator problem solving through ecological interface design. *IEEE Transactions on Systems, Man, and Cybernetics*, *25*(4), 529–545.

Vicente, K., & Rasmussen, J. (1992). Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man and Cybernetics.* 22(4), 589-606.

Wentink, M., Stassen, L. P. S., Alwayn, I., Hosman, R. J. a W., & Stassen, H. G. (2003). Rasmussen's model of human behavior in laparoscopy training. *Surgical endoscopy*, *17*(8), 1241-1246.

Wickens, C. D., Hutchins, S., Carolan, T., & Cumming, J. (2013). Effectiveness of Part-Task Training and Increasing-Difficulty Training Strategies: A Meta-Analysis Approach. *Human Factors, 55(2), 461-470.*

Wirth, J., Künsting, J., & Leutner, D. (2009). The impact of goal specificity and goal type on learning outcome and cognitive load. *Computers in Human Behavior*, *25*(2), 299–305.

Woods, D. D., & Roth, E. M. (1988). Cognitive Engineering: Human Problem Solving with Tools. *Human Factors*, *30*(4), 415–430.

Woods, D.D. (1988). Coping with complexity: the psychology of human behavior in complex systems. In: Goodstein, L.P., Andersen, H.B., Olsen, S.E. (Eds.), *Tasks, Errors and Mental Models* (pp. 128-148). London, UK: Taylor & Francis.

Woods, D. (1993). "Process-Tracing Methods for the Study of Cognition Outside of the Experimental Psychology Laboratory", in G. Klein, J. Orasanu, R. Calderwood, C. Zsambok (Eds.) *Decision Making in Action: Models and Methods* (pp. 228-251). Norwood, NJ: Ablex Publishing.

Publications from this Dissertation

The following list contains publications derived from the work in this dissertation. Parts of these publications appear in this thesis.

1. Robinson, T., Burns C. (2009). Computer Algebra Systems and Their Effect on Cognitive Load. *Proceedings of the 9th bi-annual international conference on naturalistic decision making NDM9: NDM and Computers.* London, UK. June 23-26, 2009.
2. Robinson, T., Burns, C.M. (2010). A Learning Process: Old Strategies, New Tools. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* 54, 384-388.
3. Robinson, T., Burns, C.M. (2013, in press). Focused Learning: Control of Cognitive Load in Instructional Material for a Computer Algebra System. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 55.*

Appendix A
Math 137 Topics

The following is a list of topics from the Math 137 Curriculum (University of Waterloo, 2009):

Functions, graphs, absolute values, one-to-one functions and inverses

Exponential and trigonometric functions and their inverses

Limits: formal definition, rules and examples

Continuity and intermediate values

Derivatives: definition, formulas, rules, and applications

The chain rule, implicit differentiation, applications

Extreme values, the mean value theorem

Monotone functions and curve sketching

L'Hopital's rule s

Tangent lines, Newton's method

Riemann integral, the fundamental theorem of calculus

Area problems

Error estimates for the linear approximation

Appendix B
Expert Maple User Study - Strategies Used

Table 94 shows a summary of the strategies used by the expert Maple users as discussed in section 3.3.3. A red cell background indicated the problem was solved using Maple; a blue background indicates that paper and pencil was used. The one purple cell indicates that the problem was done on paper, even though the participant was allowed to use Maple for that problem (he then redid the calculations in Maple, but noted that it was easier for him to solve the problem without Maple). Each cell shows a brief sequence of steps used to solve the problem, with a brief explanation of probable intent or purpose for each step. A step written in italic represents a step that ended up being a dead end where the participant had to find another solution strategy. The step names or Maple command names are written in bold.

Table 94 - Strategies used in expert Maple user study.

Found on the following 4 pages. Shaded cells indicate Maple use and non-shaded cell indicate non-Maple use. Maple commands are in bold. Italics indicates steps that were incorrect or did not work as expected. At the end of each list of steps for the Maple solved problems, a list of Maple syntax features used is given in small caps.

Participant / Problem	1	2	3	4	5	6	7
1	**plot** to see area between curves **solve** to find intersections of curves **Int** to check integral **value** to evaluate integral RANGE LISTS ASSIGNMENT FUNCTION CALLS	**sketch plot** to get a sense of the area to integrate **quadratic formula** to find intersections of the curves **integration** to find area **calculator** to evaluate integral	**fsolve** to find intersections of curves **int** to evaluate integral **plot** to check ASSIGNMENT RANGE EQUALITY FUNCTION CALLS	**complete the Square** to find intersection points ***substitute into original equation*** *to check end points, found the endpoints he computed were wrong* **quaratic formula** try to find endpoints again and check previous **setup integration** **calculator** to evaluate integral	**sketch plot** to see what area between curves looks like ***tried factoring*** *to find intersection points* ***calculator to plot function*** *to find intersection points* **quadratic formula** to find intercept points **setup integral** **calculator** to evaluate integral	**plot** to see what area between curves looks like **solve** to find intersection points **int** to evaluate integral ASSIGNMENT EQUALITY RANGE	sketch on paper enter equation for vertical displacement **convert** – to convert to radians ASSIGNMENT RANGE EQUALITY

Participant Problem	1	2	3	4	5	6	7
2	**compute derivative** find extremums **factor** find extremums **2nd derivative test** to determine which are min and max	***Look at context menu*** *to find minimize* ***diff*** *from context menu to get derivative* **help** look up minimize in help system **minimize** **test minimize on another function** to determine how minimize works (participant was not sure if minimize gave the minimum or location of minimum) FUNCTION CALLS	**find derivative** to find extremums **factor** to find extremums **quadratic equation** to find that two roots are complex **evaluate** to determine value at x=0, the minimum	**minimize** to find minimum of function FUNCTION CALLS	**diff** to find derivative to find extremums **solve** to find extremums **plot** to visualize where the minimum is **select and remove** to get one real solution **eval** to compute value of function at x = 0 ASSIGNMENT EQUALITY RANGE	**compute derivative** to find extremums **factoring** to find extremums **quadratic equation** to find that two roots are complex **evaluate function** to find value at x=0, the minimum	**compute derivative** to find extremums **try factor** to find extremums **quadratic equation** to find that two roots are complex **evaluate function** to find value at x=0

Participant Problem	1	2	3	4	5	6	7
3	**converting to complex numbers** to find vector sum **converted angles to radians** (but did not need to) **Add Complex numbers** **Compute magnitude and argument of complex numbers** to get final form of answer **quick check against drawing** to see if answer made sense	**compute horizontal and vertical components of each force** **convert to radians** because Maple trigonometry commands use radians **help for atan and arctan** to find proper command name **atan and sqrt** to find magnitude and angle of final vector FUNCTION CALLS	**compute horizontal and vertical components** **units package** to allow for degrees in sin and cos **help for atan and arctan** to find command name **atan and sqrt** to find magnitude and angle FUNCTION CALLS LISTS ASSIGNMENT	**Pythagorean theorem and ratios of forces** to compute resultant force vector FUNCTION CALLS	**compute horizontal and vertical components** **convert to radians** **add components** **atan and sqrt** to find magnitude and angle of resultant force **plot** to check if the answer made sense FUNCTION CALLS EQUALITY ASSIGNMENT	**write in vector Cartesian notation** **add vectors** to find resultant vector **convert back to polar**	sketch – to convert to Cartesian write in vector Cartesian notation add vectors sketch – to convert to polar convert back to polar

Participant Problem	1	2	3	4	5	6	7
4	**considered deriving projectile equation** **used textbook** to find range equation **used range equation** **then redid same computation on maple** FUNCTION CALLS EQUALS ASSIGNMENT	**drew sketch of situation** **wrote down equation for vertical displacement** ***quadratic formula*** *to find impact time* **factor** to find impact time **multiply time and initial horizontal velocity** to find range	**drew sketch of situation** **solved 2nd order differential equation** to get vertical displacement equation **factor** to find impact time **multiply time and initial horizontal velocity** to find range	**drew sketch of situation** **wrote down equation for vertical displacement** **factor** to find impact time **multiply time and initial horizontal velocity** to find range	**drew sketch of situation** **solved 2nd order differential equation** to get vertical displacement equation **factor** to find impact time **multiply time and initial horizontal velocity** to find range	**drew sketch of situation on paper** **solve 1st order differential equation** to get vertical displacement equation **solve** to find impact time **enter equation for horizontal displacement** **eval** at time of impact to calculate range FUNCTION CALLS ASSIGNMENT	sketch on paper enter equation for vertical displacement **convert** to convert to radians **solve** to find impact time **multiply** to find range EQUALS FUNCTION CALLS ASSIGNMENT

Appendix C

Maple Instructional Study - Familiar Math Instructional Material

Intersection of curves

Problem One

Problem One: solve for the intersection points of a line and a parabola. Draw a plot showing the intersection points then find x values of the intersection point(s).

Find the intersection of the line

$y = 2\,x + 3$

and the parabola

$y = 3\cdot x^2 - 4$

Entering Expression

In maple you can assign an expression to a variable.

For example, on the next line type f := 2*x+3 and press enter

```
>
```

To see that f has been assigned the expression enter f on the next line and press enter.

```
>
```

Similarly, you can assign the expression $3\cdot x^2 - 4$ to the variable g by typing g := 3*x^2-4 (after typing the 2 in the exponent press the right cursor key to exit the exponent before typing -4)

```
>
```

Type g and press enter on the next line to see the value of g

```
>
```

Plotting functions

Plotting a Single Function

In Maple you can draw the plot of a function using the plot command.

To plot the function x^2 enter plot(x^2,x= -10..10) on the next line

```
>
```

You can plot one of the functions you previously assigned to a variable.

For example, to plot the line enter plot(f,x=-10..10) on the next line and press enter.

```
>
```

Plotting Multiple Functions

To plot both the line and the parabola on the same plot, you can pass the plot command a list of

expressions. To plot both functions enter
plot([f,g],x=-10..10)

>

You probably want to zoom in on the region containing the intersection points. Try modifying the previous command to plot the region from x equals -3 to 3.

Solving Equations

You would now like to find the x values of the intersection points of the line and parabola. This is found by determining what x values give the same y values for the two equations. So we need to create an equation where the left hand side is f and the right hand side is g.

Enter f = g on the next line and press enter.

>

You need to solve this quadratic for x. This can be done using Maple's solve command.

On the next line enter eqSolve(f=g,x)

>

This solution shows the exact values of the two solutions to $2x + 3 = 3x^2 - 4$.

To get floating point values you can use Maple's evalf command.

Add a r := to the start of the previous command to give r := eqSolve(f=g,x) and press enter again. This will allow you to refer to the solution with the variable r.

Next, on the line below, type evalf(r) and press enter.

>

You now have the solution to the problem in exact and floating point form. These are the x values of the two intersection points.

Evaluating an Expression

To find the value of an experession at a point you can use the eval command.

To find the value of $2x + 3$ at the first intersection point, type eval(f,x=-1.230138587) on the next line

>

Find the value of $2x + 3$ at the second intersection point

>

Problem Two

Problem two: plot and find the intersection of two parabolas.

Create a plot showing the intersection of the two parabolas

Find the y values of the intersection points

$y = 3 \cdot x^2 - 3$
$y = -2 \cdot x^2 + 4$

Note: to insert a new input line below the current line press crtl-j (meta-j on a mac)

>

Finding a Derivative and Tangent Line

Problem Three

We can find the derivative of a function with the *diff* command

Assign f the expression sin(x)

>

Plot f from 0 to 2

>

Now type g := diff(f,x) to assign g the derivative of f

>

Solve, using eqSolve, g=0 to find a point where the tangent line is horizontal (slope of 0) and assign this value to t

>

Type evalf(t) to get a floating point value for t

>

Plot the functions f and y=1 from 0 to 2 to see where the line with slope 0 touches the curve.

>

Which of the above values of t is shown in this plot?

Problem Four

Find a point on the curve sin(x) where the derivative is 1

>

Plot sin(x) and x from -2 to 2 to see the tangent line with slope 1

>

Appendix D
Maple Instruction Study - Unfamiliar Math Instructional Material

Approximation of Functions

Problem One

A function, such as sin(x) can be approximated using a line or quadratic over a short distance.

Entering an Expression

In maple you can assign an expression to a variable.

For example, on the next line type f := sin(x) and press enter

```
>
```

To see that f has been assigned the expression enter f on the next line and press enter.

```
>
```

Plotting an Expression

Plotting a single function
In Maple you can draw the plot of a function using the plot command.

To plot the function x^2 enter plot(x^2,x= -10..10) on the next line (after typing the 2 in the exponent press the right cursor key (right arrow key) to exit the exponent before typing ,)

```
>
```

You can plot one of the functions you previously assigned to a variable.

For example, to plot the line enter plot(f,x=-10..10) on the next line and press enter.

```
>
```

Derivative of a Expression

You can find the derivative of a expression with the *diff* function.

On the next line type g := diff(f,x) and press enter

```
>
```

You can find the value of an expression at a point with the eval command

Type eval(g,x=1.2) to find the value of the derivative at 1.2

```
>
```

Approximation of a Function

You can approximate a function with a line that will be close to the original function for a short distance.

The formula for an approximation of the function f(x) at x_0 is

$l(x) = f(x_0) + \frac{d}{dx} f(x_0) \cdot (x - x_0)$

to form an expression for the approximation of sin(x) at 1.2 enter
$l := eval(f, x = 1.2) + eval(g, x = 1.2) \cdot (x - 1.2)$ on the next line and press enter

```
>
```

Plotting Multiple Expressions

To plot both the original expression and the approximation on the same plot, you can pass the plot command a list of expressions. To plot both functions enter
plot([f,l],x=-10..10)

```
>
>
```

To zoom in on the y values of interest we can change the above command to plot([f,l],x=-0.5..2,y=-2..2).

Quadratic Appoximation

You can also approximate a function with a quadratic function.

The formula for a quadratic approximation of f(x) at x_0 is:

$$q(x) = f(x_0) + \frac{d}{dx} f(x_0) \cdot (x - x_0) + \frac{1}{2} \cdot \frac{d^2}{dx^2} f(x_0) (x - x_0)^2$$

You first need the second derivative of f(x), which you can find by differentiating g (which you should assign to h):

```
>
```

Next we can create the expression for q

> $q := eval(f, x = 1.2) + eval(g, x = 1.2) \cdot (x - 1.2) + \frac{1}{2} \cdot eval(h, x = 1.2) \cdot (x - 1.2)^2$

Now plot the f, l, and q on the same plot

```
>
```

You can see that the quadratic approximation is accurate close to 1.2

You can compute and plot the error of the approximation with

> $d := |f - q|$

> $plot(d, x = -0.5..4)$

You can see that after a while the error increases monotonically.

Solving Equations

You can find the points at which the error is equal to 0.1

the eqSolve command will find approximate solutions to equations. You can find the real solutions with the eqSolve
command:

```
> eqSolve(d = .1, x)
>
```

Problem Two

You will now investigate the function ln(x) and its quadratic approximation at x=2

1. Plot the function ln(x) from 0.1 to 5
2. Assign the first and second derivatives of f to g and h respectivly.
3. Assign the expression for the quadratic approximation to q.
4. Plot the original function and the quadratic approximation on the same plot
5. Assign d the error between f and q.
6. Plot the error and find where the error is 0.1

Note: to insert a new input line below the current line press crtl-j (meta-j on a mac)

Appendix E

Maple Instructional Study – Test

Test - Design of Instructional Material for Mathematical Software

Question 1

Add a new input line to a document below this line.

Question 2

What goes in place of XX in the following command to create a plot from -2 to 2?

```
plot(sin(x),XX)
```

a) -2 .. 2
b) x=-2..2
c) x=2,2
d) 2,2
e) none of the above

Answer:

Question 3

```
f := sin(x)
diff(f,x)
```

returns:

a) an expression for the derivative of f
b) the value of the derivative at 0
c) a numerical value
d) none of the above

Answer:

Question 4

What goes in the place of XX in the following plot command to create a plot of functions f and g from 1 to 3?

```
f := ln(x)
g:=sin(x)
plot(XX,x=1..3)
```

a) f,g
b) {f,g}
c) (f,g)
d) [f,g]

e) none of the above

Answer:

Question 5

Which command assigns sin(x) to f?

a) f = sin(x)
b) f := sin(x)
c) f <- sin(x)
d) f == sin(x)
e) none of the above

Answer:

Question 6

Which command will give you the value of sin(x) at 3?

a) eqSolve(sin(x),x=3)
b) eqSolve(sin(x),3)
c) eval(sin(x),x=3)
d) eval(sin(x),3)
e) none of the above

Answer:

Question 7

What command will find the point where x^3 is 17?

a) $eval(x^3, 17)$
b) $eval(x^3, x = 17)$
c) $eqSolve(x^3, x = 17)$
d) $eqSolve(x^3, 17)$
e) $eqSolve(x^3 = 17, x)$
f) $eqSolve(x^3 = 17)$

Answer:

Question 8

Which command plots sin(x) from 1..10?
a) plot(sin(x),1..10)
b) plot(sin(x),1,10)
c) plot(sin(x),x,1,10)
d) plot(sin(x),x=1..10)
e) none of the above

Answer:

Question 9

Tell the experimenter the sequence of commands to solve the equation x^2+1=4

Question 10

Tell the experimenter the sequence of commands to find a point where the derivative of sin(x)+cos (x) is 0

Question 11

Find the derivative of the function tan(x)

```
>
>
>
```

Question 12

Create a plot of the function sin(x) from -1 to 1?

```
>
```

Question 13

Create one plot of the functions sin(x) and cos(x) from -2 to 2?

```
>
```

Question 15

Find the x value of the point on the function ln(x) where the derivative is 0.11

```
>
>
>
```

Appendix F
Maple Instructional Study - Test Results

Table 95 shows the test results for the Maple instructional study. The results are discussed in section 5.1.3.1. There were originally 15 questions, but the 14th was removed when I discovered, after the third participant, that the familiar condition material did not cover the concept needed. For some of the questions, answers were marked correct only if the participant could complete the problem without making corrections to previous work (as discussed in section 4); those question numbers are bolded in the table.

Table 95 - Test Result for Maple Instruction Study.

1 indicates a correct answer and 0 indicates an incorrect answer. The bolded question were marks right or wrong if the participant completed the question without backtracking.

Participant	Question Condition	1	2	3	4	5	6	7	8	**9**	**10**	**11**	12	13	**15**	Total
1	unfamiliar	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14
2	unfamiliar	1	1	1	1	1	1	0	1	1	1	1	1	1	0	12
3	familiar	1	1	1	1	1	1	0	1			1	1	1	1	11
4	familiar	1	1	0	1	1	0	1	1	0	1	1	1	1	1	11
5	unfamiliar	0	1	0	0	1	1	0	1	0	0	1	1	1	1	8
6	familiar	1	1	1	1	1	0	1	1	1	1	1	1	1	1	13
7	unfamiliar	0	1	1	1	1	1	0	1	0	0	1	0	1	1	9
8	familiar	1	1	1	1	1	1	1	1	1	1	1	1	1	0	13
9	unfamiliar	1	1	1	1	1	0	0	1	1	0	1	1	1	0	10
10	familiar	1	1	1	1	1	1	1	1	1	0	1	1	1	0	12
11	unfamiliar	1	1	1	1	1	1	0	1	0	0	1	1	1	0	10
12	familiar	1	1	1	1	1	1	1	1	0	0	1	1	1	1	12

Participant	Question Condition	1	2	3	4	5	6	7	8	**9**	**10**	**11**	12	13	**15**	Total
13	familiar	1	1	1	1	1	1	0	1	0	0	1	1	1	0	10
14	unfamiliar	1	1	1	1	1	0	0	1	1	0	1	1	1	0	10
15	familiar	1	1	1	1	1	1	1	1	0	0	1	1	1	0	11
16	unfamiliar	1	1	0	1	1	1	0	1	1	0	1	1	1	0	10
17	familiar	1	1	1	1	1	1	1	1	1	0	1	1	1	0	12
18	unfamiliar	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14

Appendix G
Maple Instructional Study - NASA-TLX Results

Table 96 - NASA-TLX results for Maple instruction study.

The NASA-TLX was not completed by participant 1.

Participant	Condition	TLX Instructional Material	TLX Test
1	unfamiliar		
2	unfamiliar	48.7	48.6
3	familiar	58	48.8
4	familiar	19.7	54.1
5	unfamiliar	42.3	46.7
6	familiar	53.4	30.6
7	unfamiliar	26.7	61.3
8	familiar	33.9	37.9
9	unfamiliar	46.7	53.8
10	familiar	23.3	20.6
11	unfamiliar	55.1	66.8
12	familiar	82.9	53.5
13	familiar	51.2	42.6
14	unfamiliar	33	55.8
15	familiar	56.8	59.9
16	unfamiliar	35.3	67.8
17	familiar	46.7	48.4
18	unfamiliar	6.67	6.86

Appendix H

Logo Study – Pre-Study Survey

Participant Number _________

Sex   Male / Female

Age   _____

Current program and Degree ________________________

Year/Term ________

1. How many programming/computer science courses did you take in high school?

2. What programming/computer science courses have you taken during your undergrad? Include and circle any courses that you are currently taking.

1. Mark where you feel your programming experience lies on the following scale:

No Experience	Some Experience	High Level of Experience	Expert

4. Have you used the programming language Logo before? If so, please indicate when you used it and for how long.

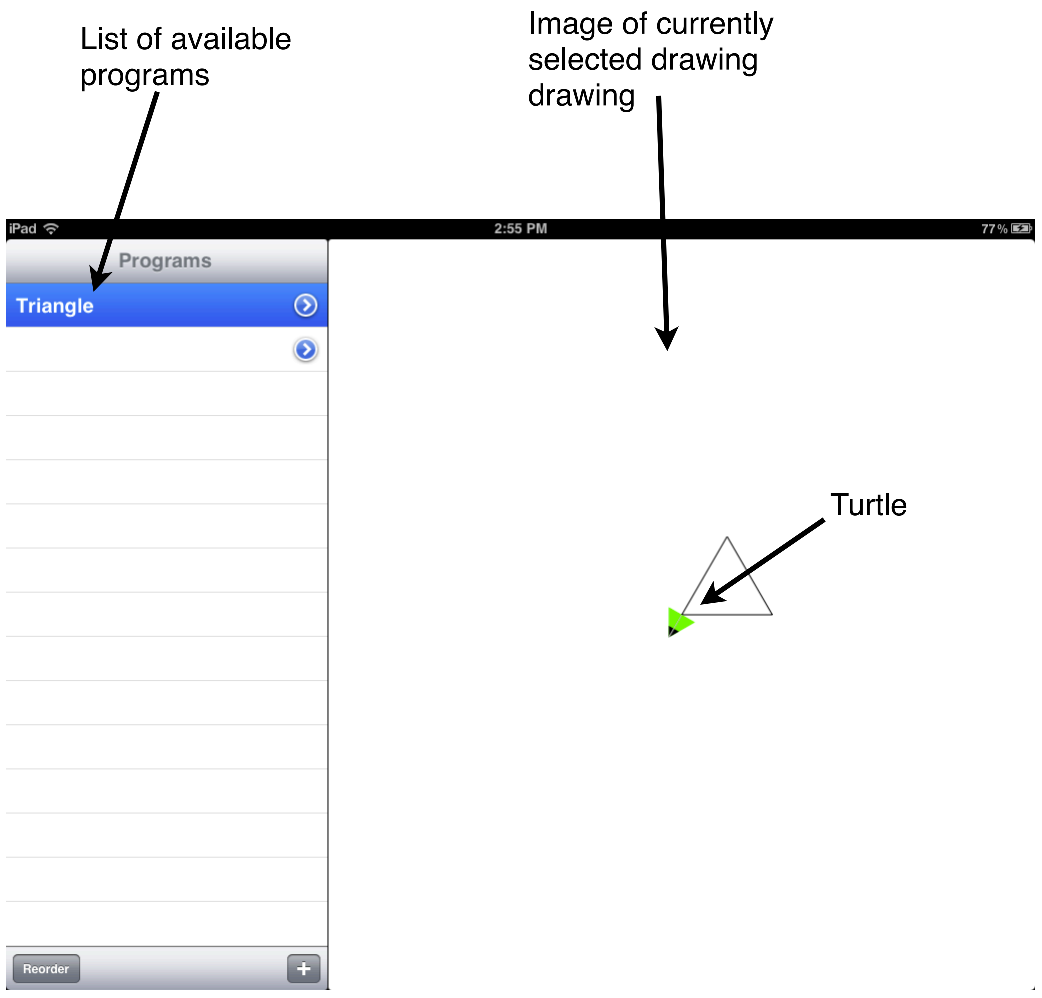
5. Have you used Lego mindstorms or a similar system before? If so, please indicate when you used it and for how long.

Appendix I

Logo Study – Introduction to Logo Instructional Material

Main Screen

The main screen that you see when you first start Logo looks like this:

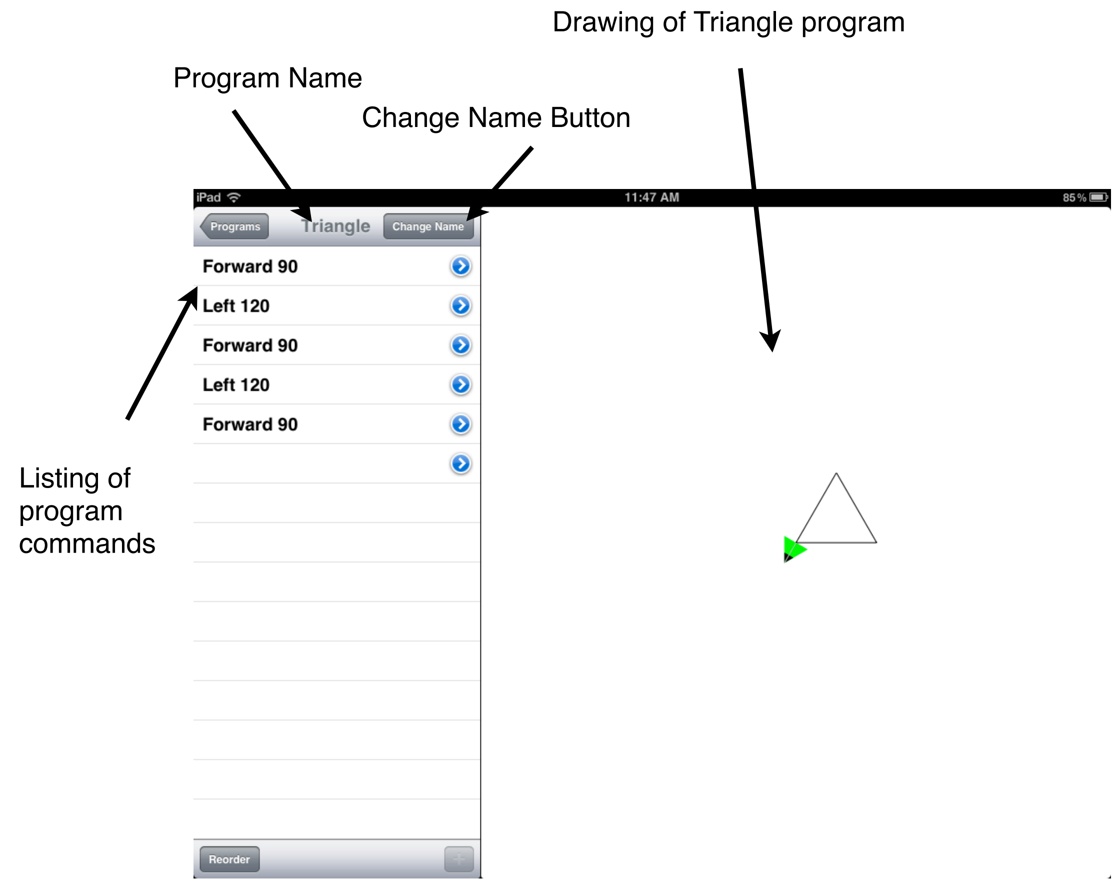

List of available programs
Image of currently selected drawing drawing
Turtle
iPad
2:55 PM
77%
Programs
Triangle
Reorder
+

Logo allows you draw images by issuing a number of commands to a turtle to create a program. There is one program already defined, and it is shown in the program list on the left of the main screen. If you touch on the program named *Triangle* in the program list, you will be able to see the drawing that results from that program. The program consists of a number of commands that tell the turtle how to move. As the turtle moves, it draws a line behind it, showing where the turtle has been.

Viewing a program

To view the listing for an existing program, tap on the button next to the desired program.

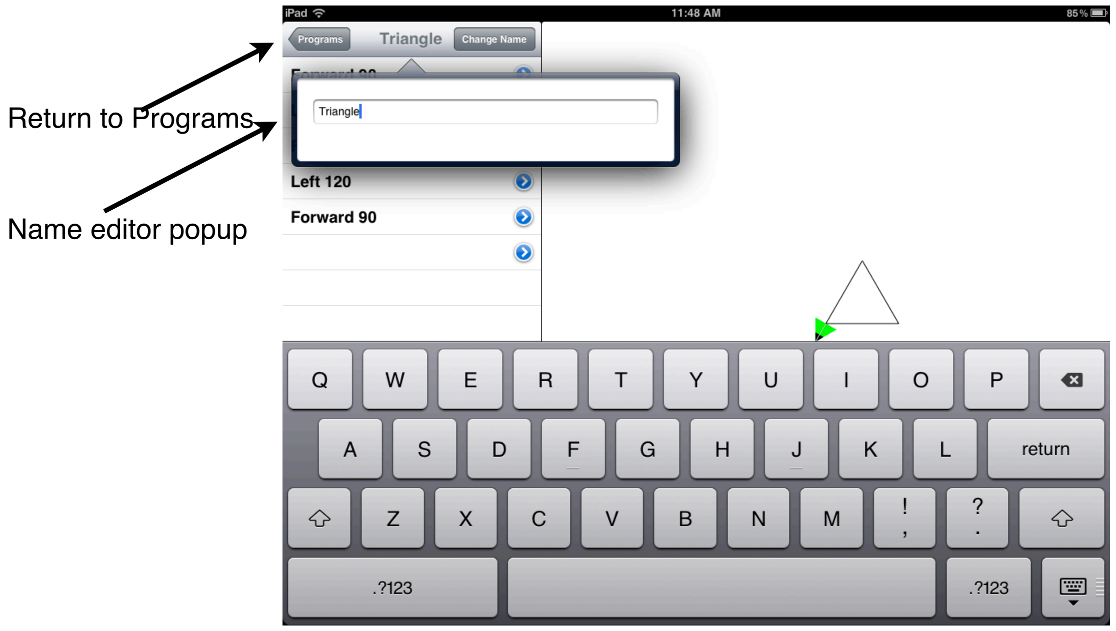
For example, if you tap the button next to the **Triangle** program, you will see the following screen. Don't worry about what the commands do yet; first you will learn what the different controls on the screen do.



```
Forward 90
Left 120
Forward 90
Left 120
Forward 90
```

Renaming a Program

You can change the name of the program by touching the **Change Name** button. Try it. You will see the following screen



You can edit the name of the program with the keyboard. Change the name of the program to *My Triangle*.   When you are done changing the name, tap somewhere outside the name editor popup (but not on the keyboard) to complete the name change.

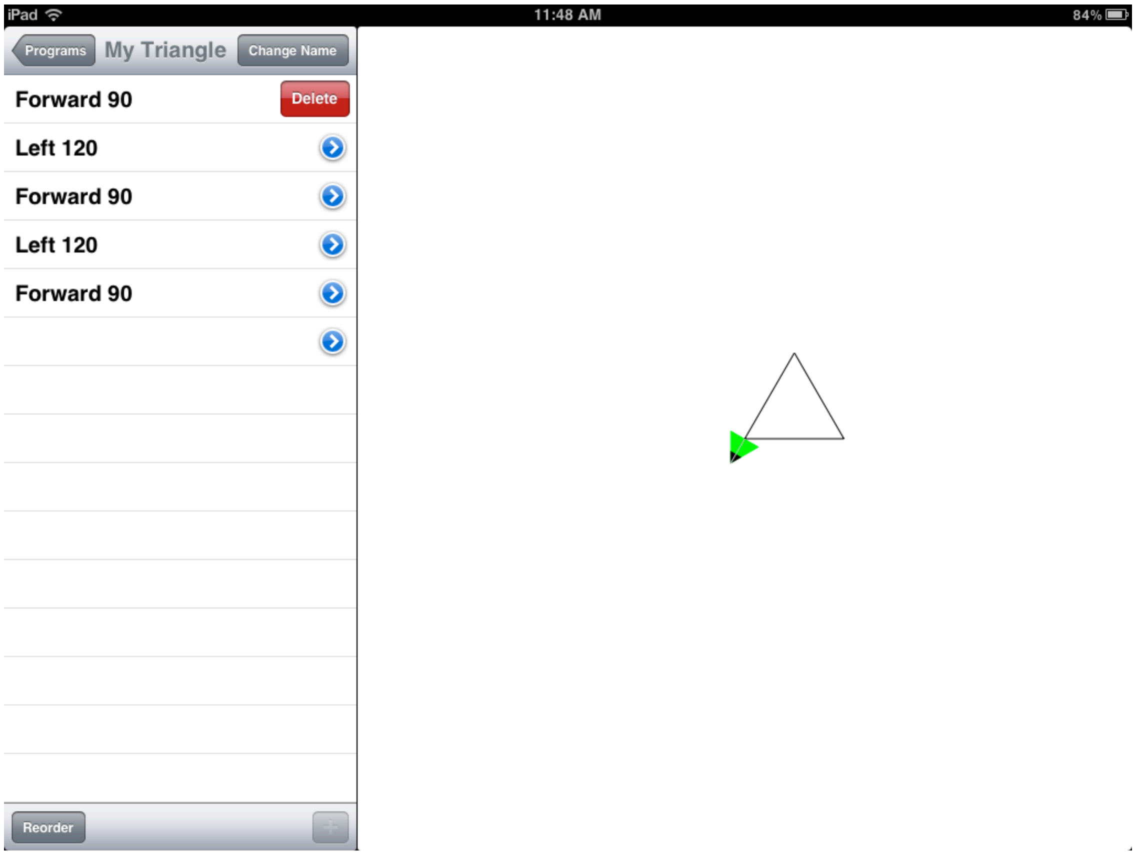
If you now touch the **Programs** button, found in the top left corner, the software will return to the main listing of programs and you will see that the Triangle program has been renamed.

Click on the ❯ next to the *My Triangle* program to display the program listing again.

Deleting a Command

To delete the first command from the *My Triangle* program, complete the following steps:

2. Put your finger down in the line with the first *Forward 90* command (but not on the button).
3. Drag your finger to the right about 1cm. A delete button should appear, as in the next figure.



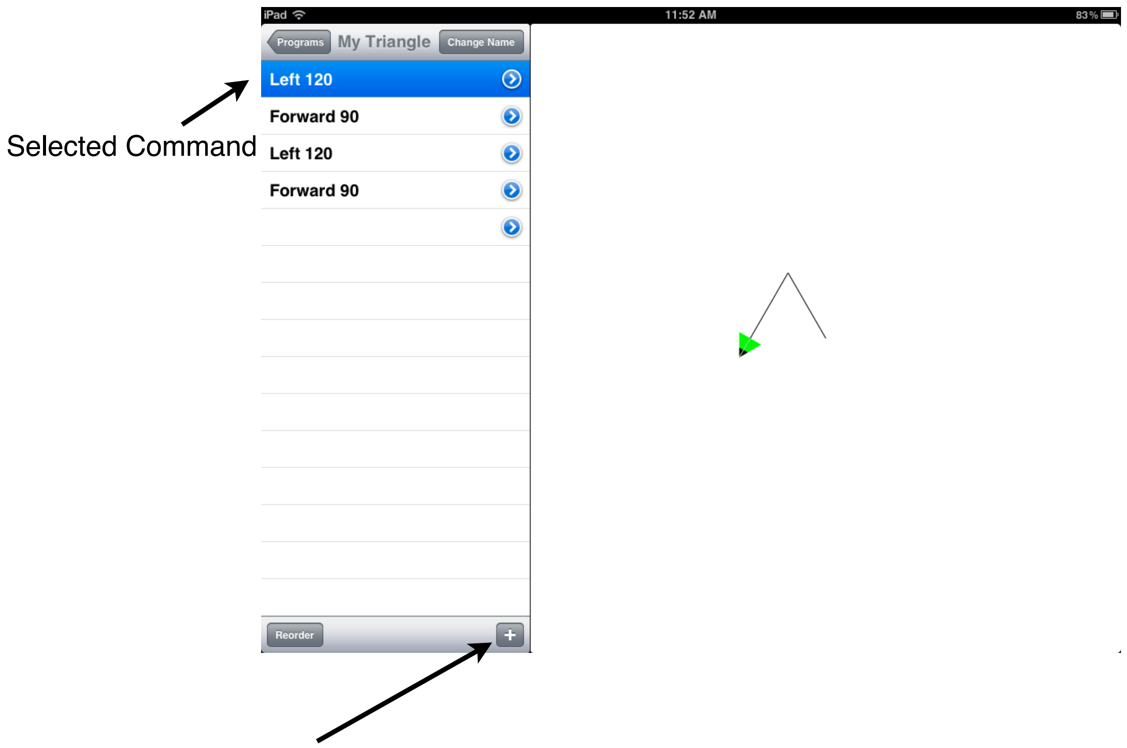
Forward 90
Left 120
Forward 90
Left 120
Forward 90

2. Touch the delete button.

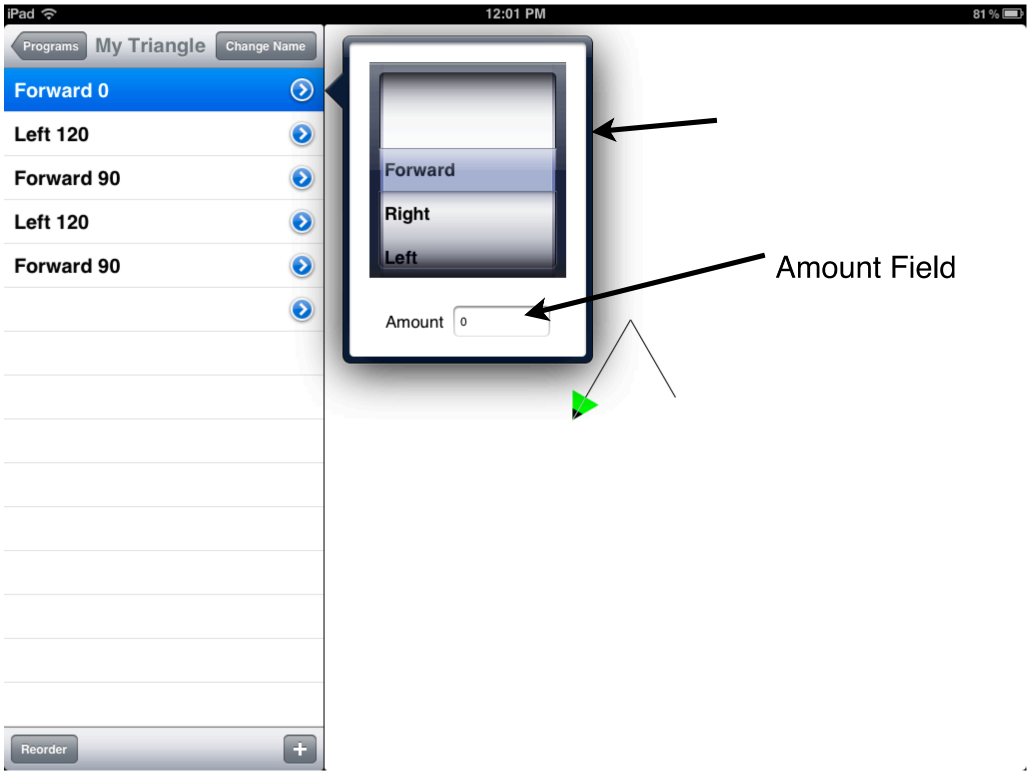
You will see that the first *Forward 90* command has been deleted.

Adding a Command

To add back the deleted forward command, tap on the line below where you want to add the command (do not tap on the button). Tapping on the line will select the command as shown.

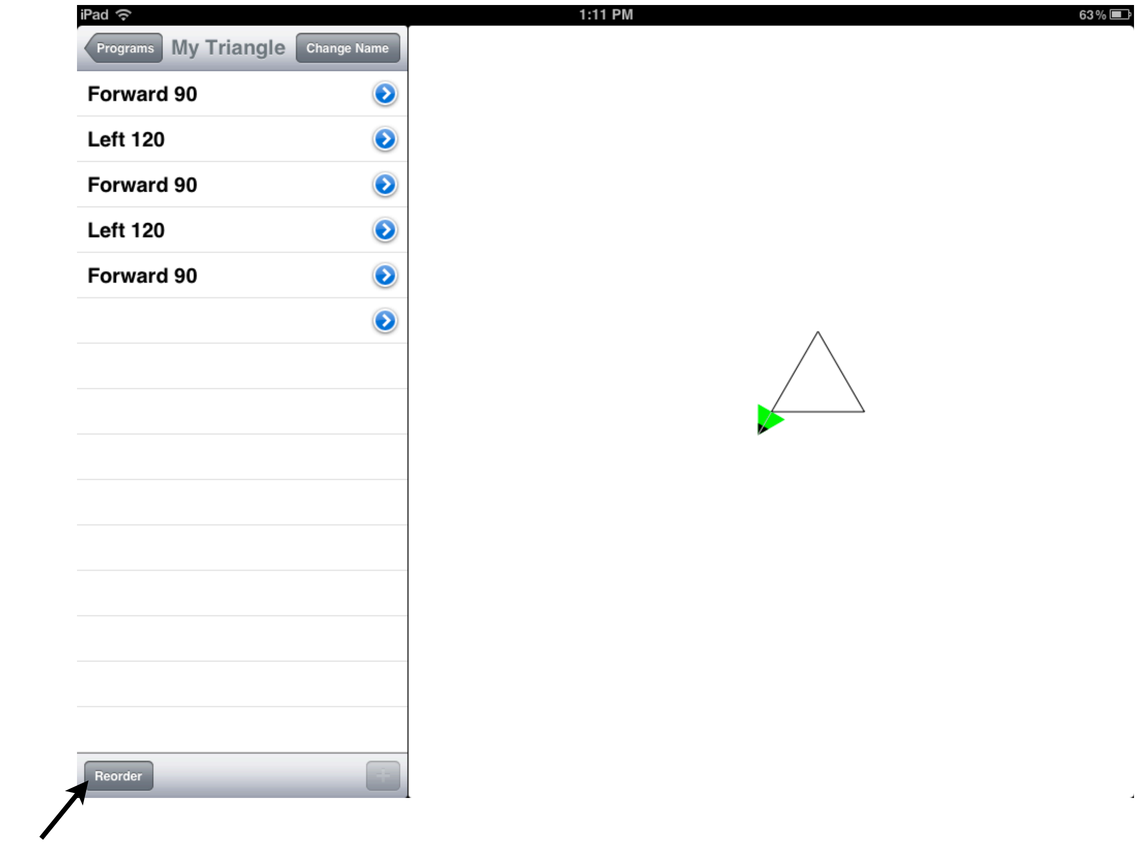


Now touch the button to add a new command above the selected command. You will see the following screen.



In this case, the *Forward* command is already selected, but you need to change the distance or amount of the *Forward* command. Touch in the text field for amount and use the keyboard that is displayed to change the amount to 90. Then touch outside the Command Editor Popup, to make the change. You will see the original triangle restored.

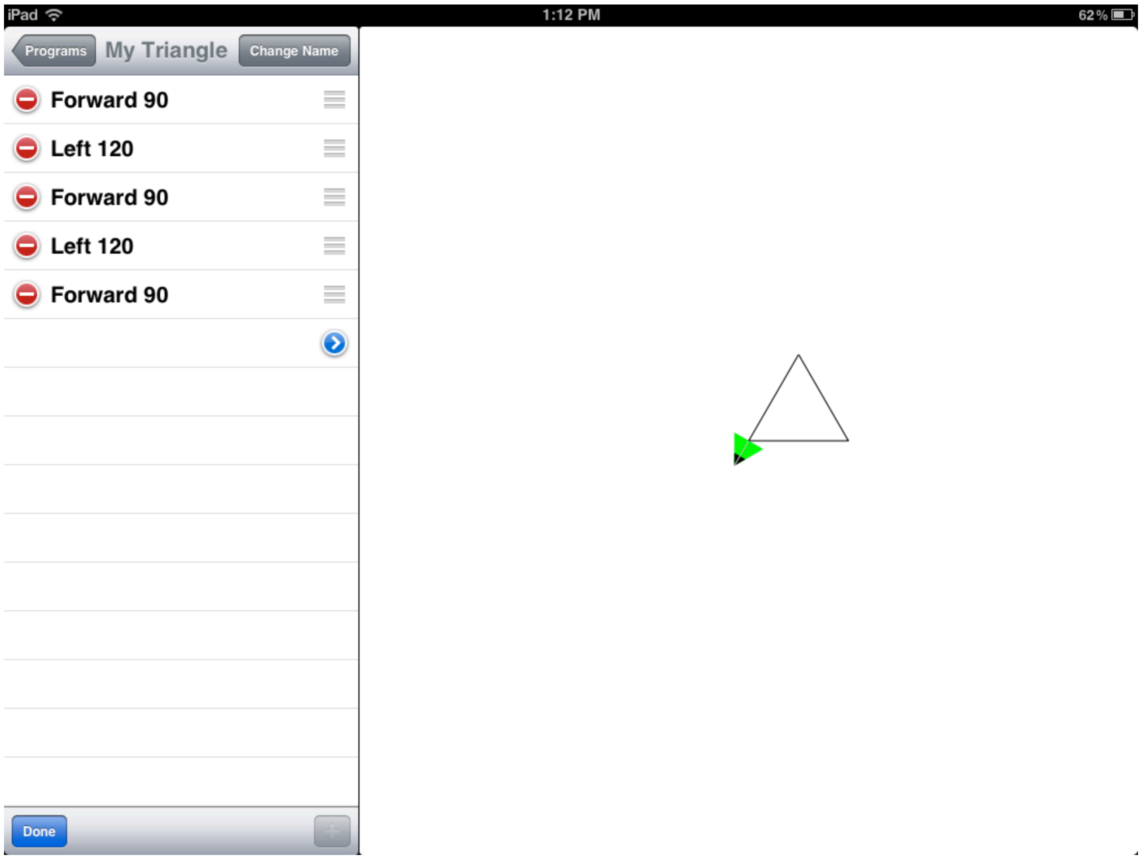
Moving a Command



Reorder Button

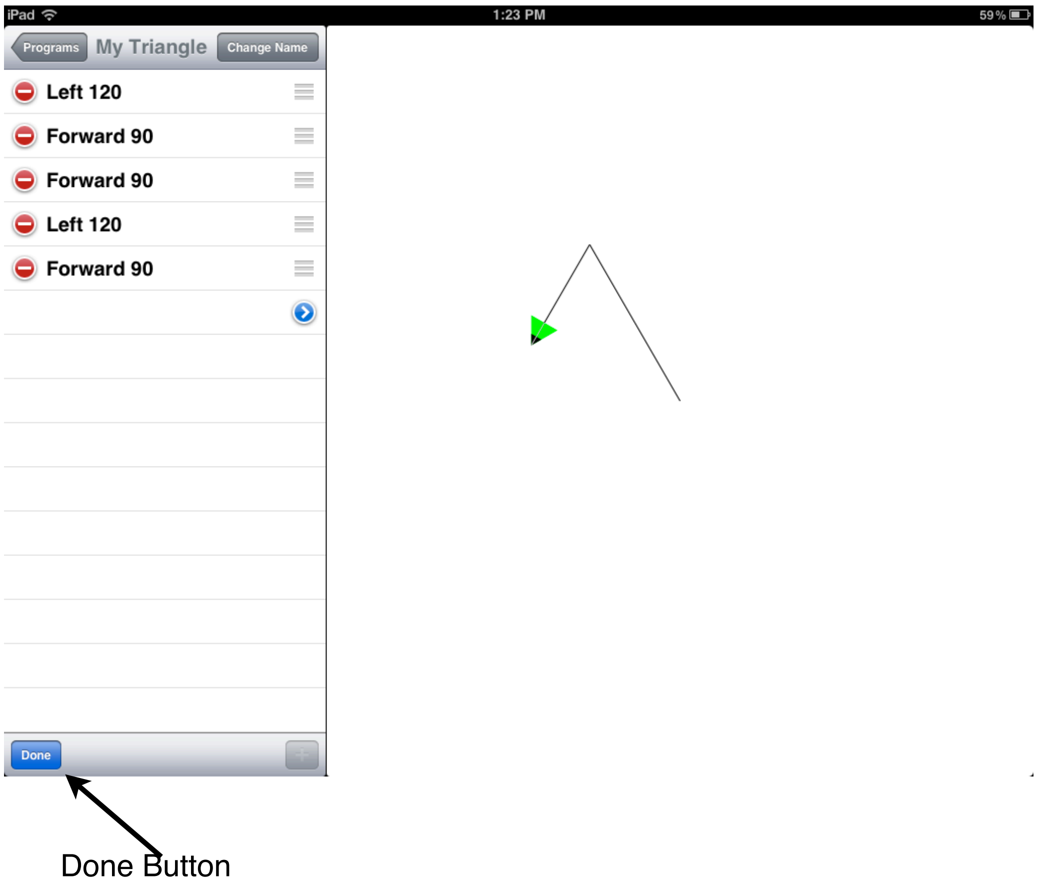
To move or rearrange commands, first touch the **Reorder** button at the bottom of the list of commands.

After touching the **Reorder** button, each command that can be moved will have a reorder control (☰) displayed on the far right of the command as shown in the next figure.



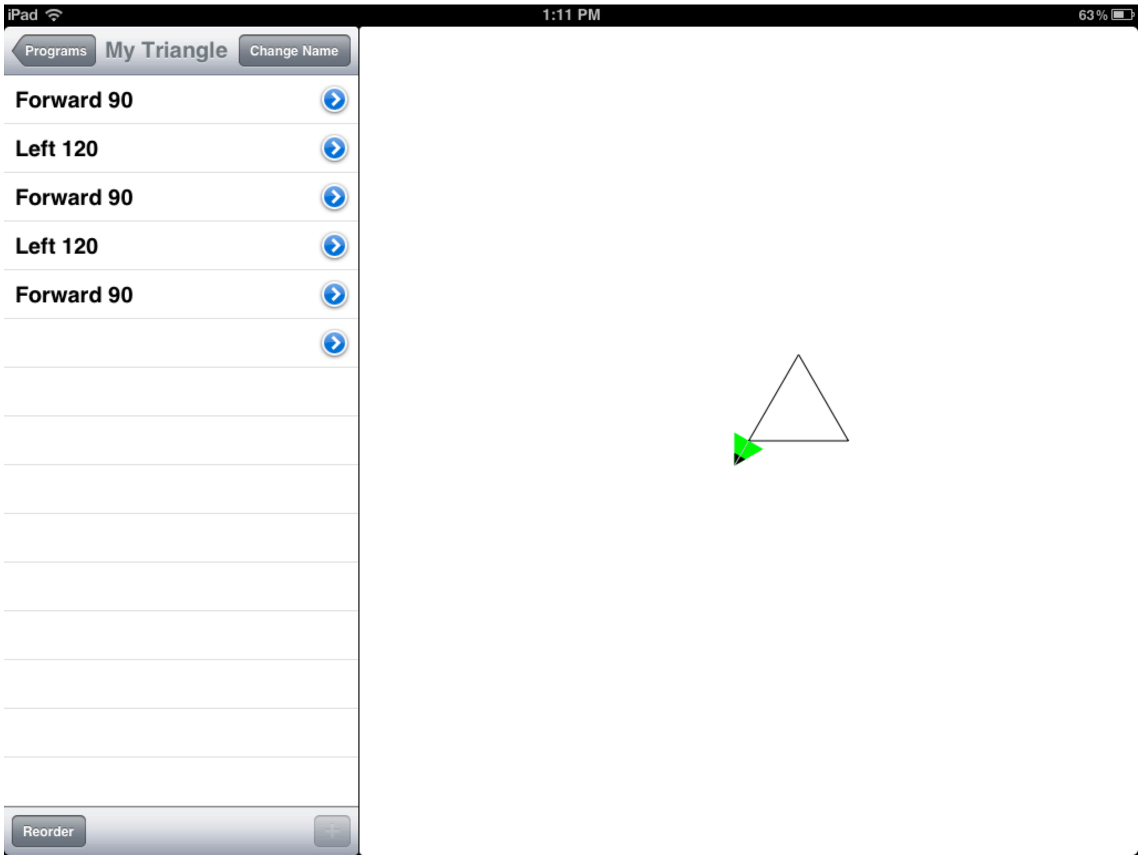
Forward 90
Left 120
Forward 90
Left 120
Forward 90

By touching and then dragging the control, the associated command can be moved. Touch the control next to the first forward command and drag it to the location below the second forward command. After removing your finger, you will see the image change to reflect the new command order.



Now drag the forward command back to its original location. When you are done moving commands, touch the **Done** button to return to the normal mode.

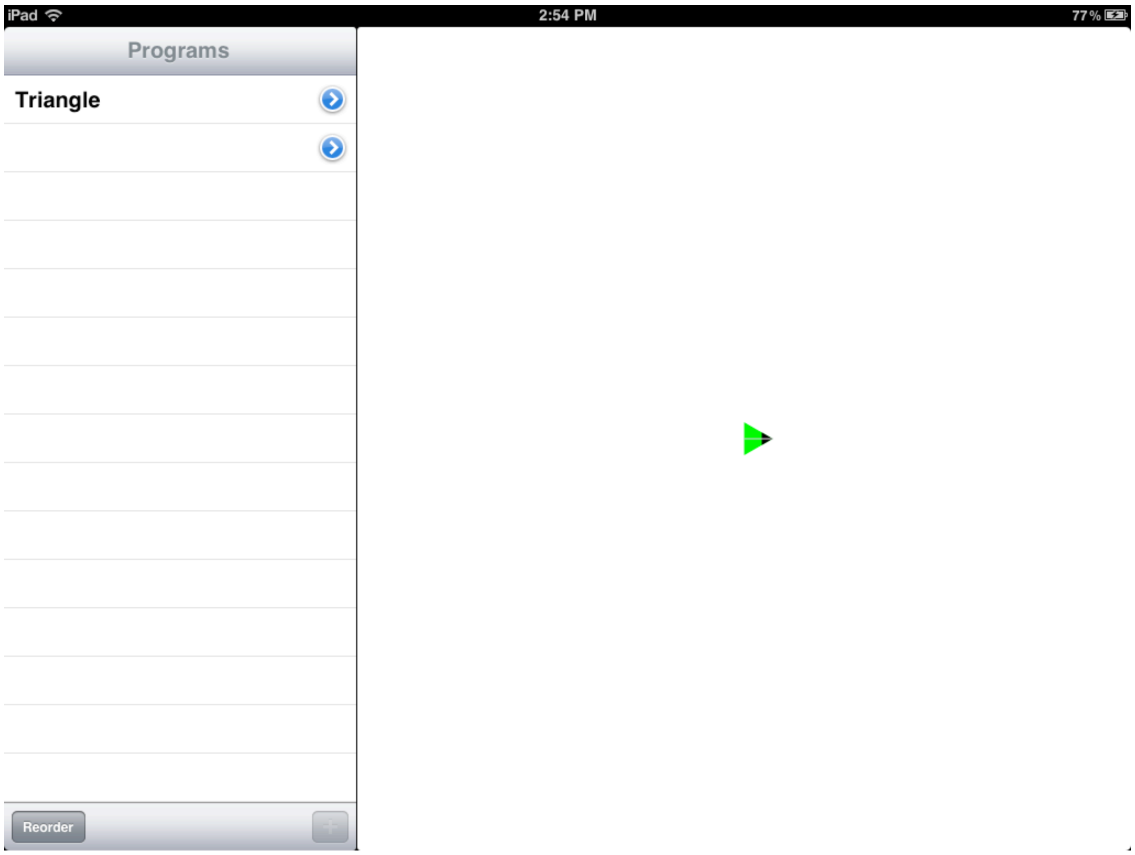
Make sure your Triangle looks the same as the one in the next diagram.



Forward 90
Left 120
Forward 90
Left 120
Forward 90

Basic Commands

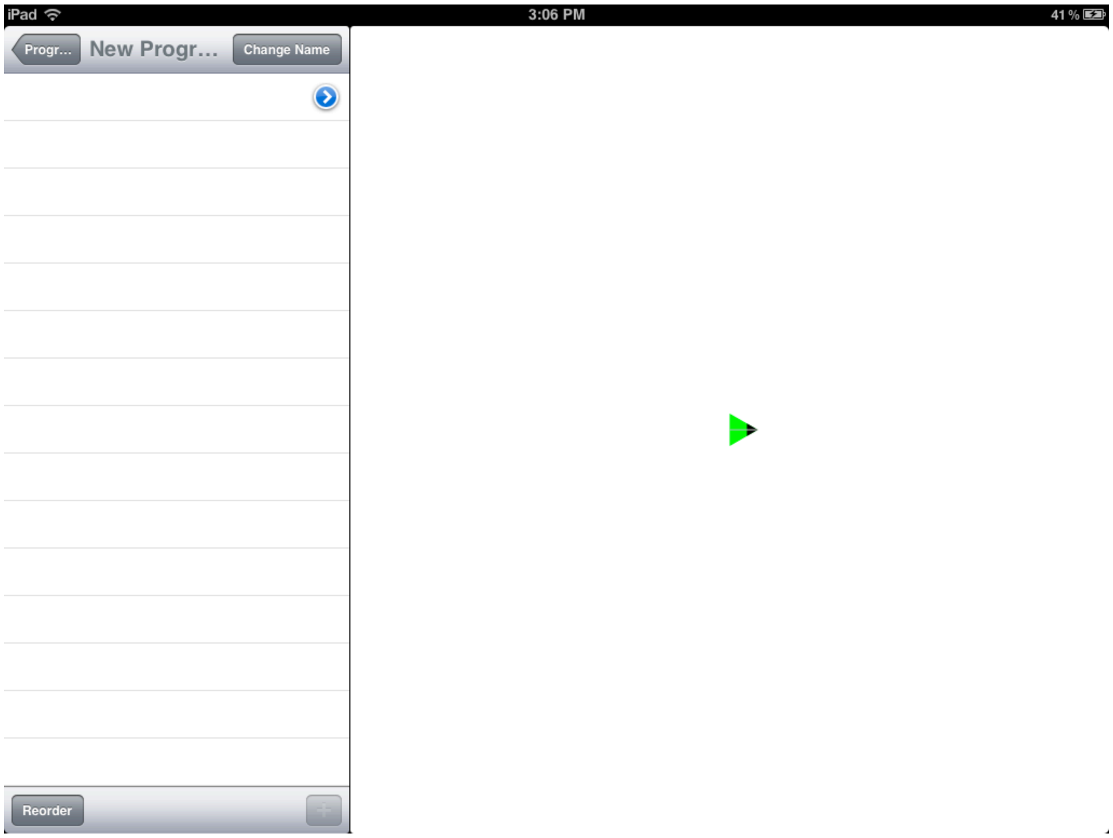
This section will introduce you to some basic commands. First return to the main programs list by touching the **Programs** button in the top left corner. Your screen should then look like the following:



Now add a new program to the program list. You can do this in one of two ways:

1. Select an existing program and touch the add + button. Then touch the button beside the new program to open it.
2. As a shortcut touch the button beside the blank program listing at the end of the program list. This action will add a new program to the end of the list and open the new program.

After performing instruction 1 OR 2 you will see the following blank program listing.

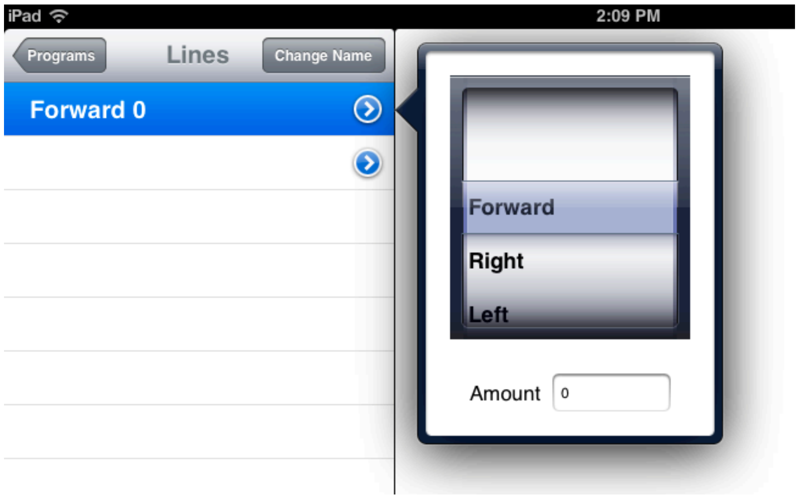


Change the name of the program by tapping on the **Change Name** button and entering the name *Lines*. Then tap outside the popup to dismiss it.

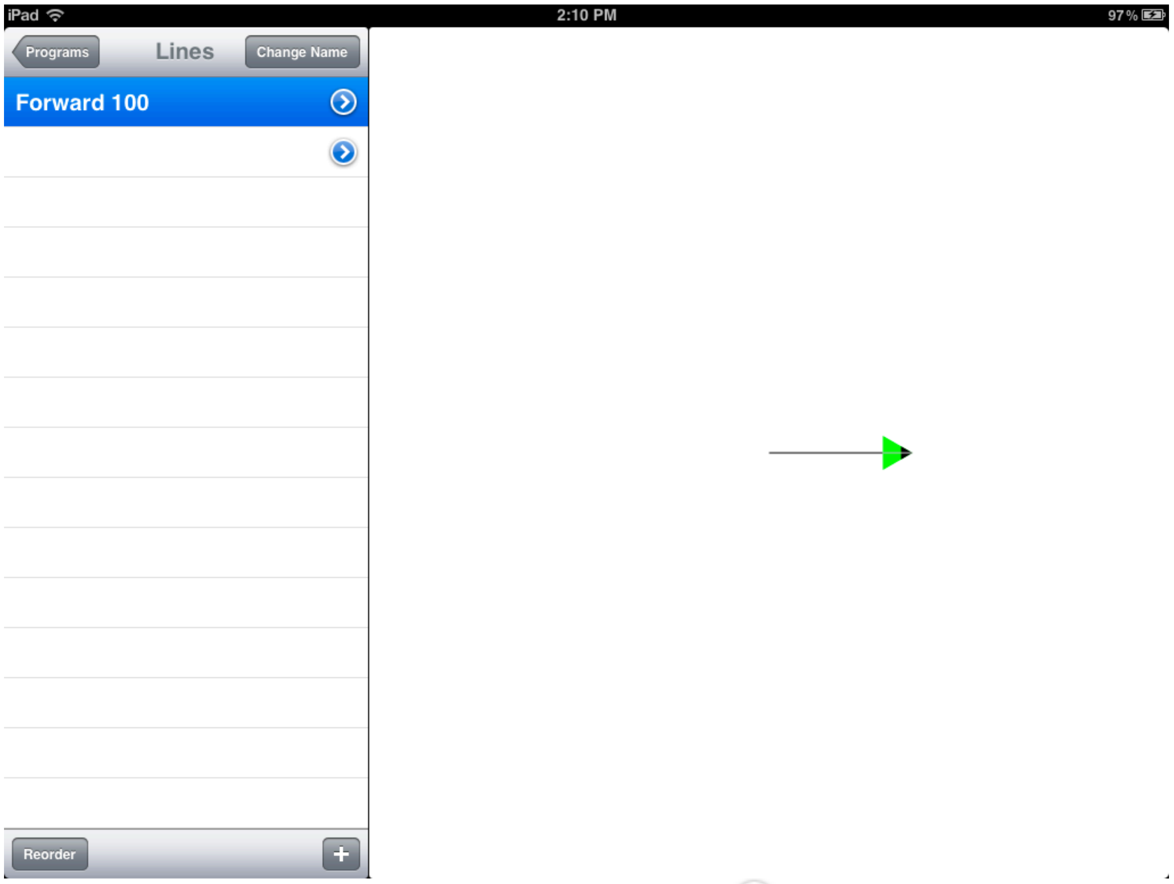
Forward Command

The forward command moves the turtle in the direction it is pointing a certain number of pixels. If the turtle's pen is down, which it is by default, the turtle will leave a line behind it as it moves. To add a command, touch the button beside the blank command at the bottom of the command list.

You will see the following popup



The forward command is already selected so you only need to tell the turtle how far to move. Tap the amount field. Use the keyboard that appears to enter a value, such as 100, in the field. Tap outside the command editor to return to the program screen and see the results of the forward command. You should see the following.



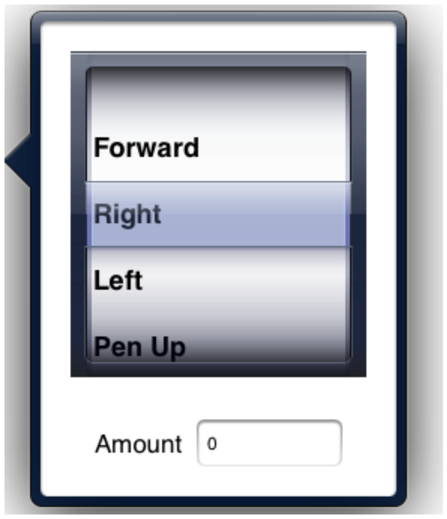
You can change the Forward command by tapping on the button next to the forward command. The command editor popup will appear. Try changing the amount for the forward command to 50. Then tap outside the command editor.

Notice the difference of the line length on the screen.

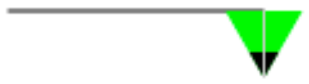
Change the line length back to 100.

Turning Commands

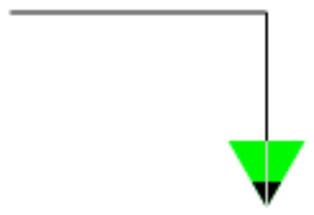
Add a new command to your program. Change the command name to *Right*, as in the figure below (sliding your finger up on the list of commands will “spin” the list and select a new command).



In the amount field, enter an angle, in degrees. For now, enter 90. Then tap outside the popup to dismiss it. You should now see the turtle is pointing down.



Add a new command to move the turtle forward 50 to create an L shape.



Finally, try changing the Right 90 angle to Right 45.

Pen Up and Pen Down

The *Pen Up* command allows you to move the turtle without drawing a line. For example if you want to start drawing at the far left of the screen, you can start your program with the commands

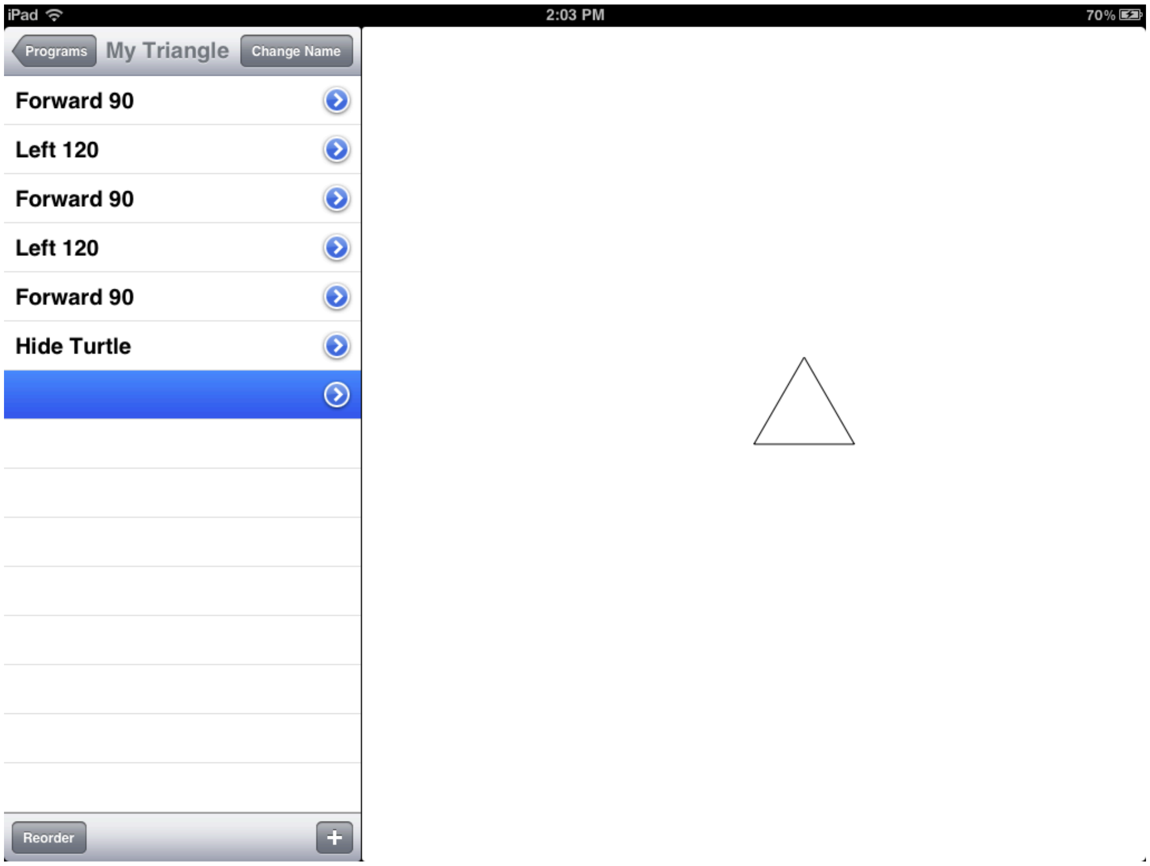
```
Pen Up
Left 180
Forward 350
Right 180
Pen Down
...
```

Create a new program called *Left Edge* and give the above a try. Then draw something on the left edge of the screen by added a few more commands.

Hide Turtle

You can add a hide turtle command to the end of a program to display the image with the turtle hidden.

For example, see the following program:



Forward 90
Left 120
Forward 90
Left 120
Forward 90
Hide Turtle

Try adding the Hide Turtle command to the end of the My Triangle program.

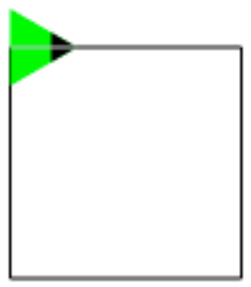
Appendix J

Logo Study - High Cognitive Load Logo Instructions for Subprograms and Repeat Command

Subprograms

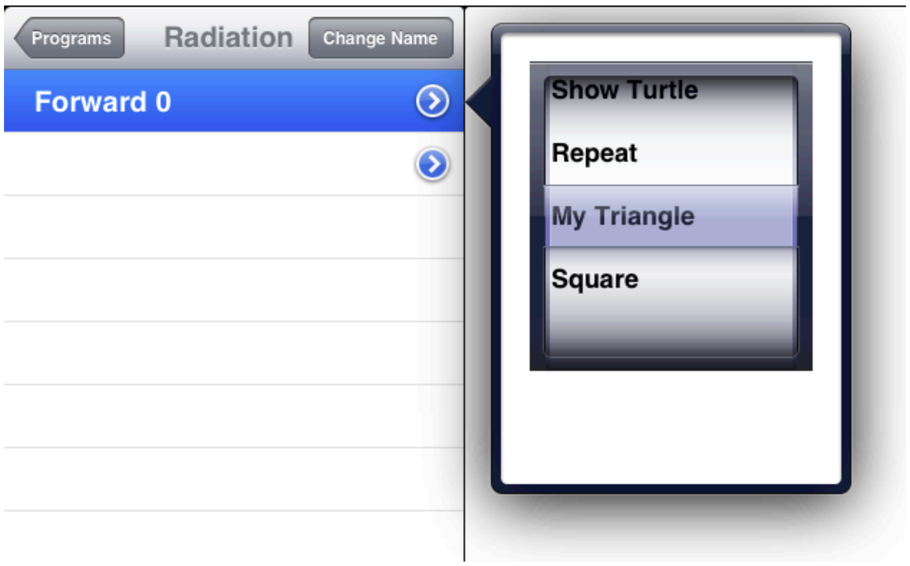
Do not use the *Repeat* command for tasks in this section.

1. Create a program called Square that draws a Square with sides 90 pixels long and ends up with the turtle in the position and with the direction shown.

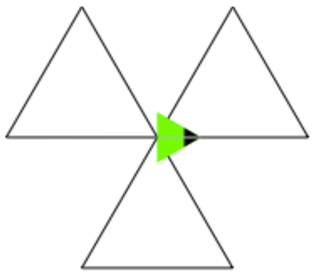


2. Create a program called Pentagon. Create a 5 sided figure where each side is length 100 and each angle is a right turn of 72 degrees.

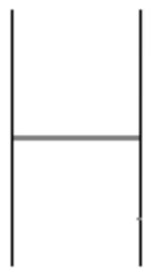
3. You can reuse existing programs to create a new program. Create a program called *Radiation*. You can use the *My Triangle* program by selecting it as the name for a command as shown:



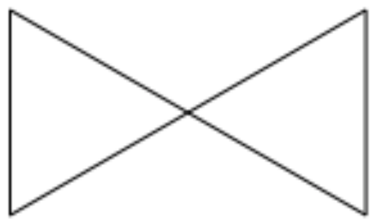
Use the My Triangle program to create the following image



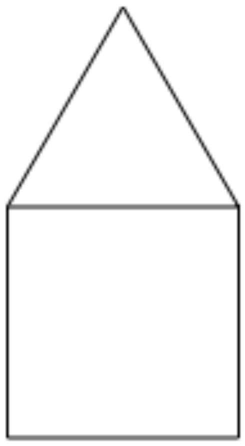
4. Create a new program named H. Setup the program to draw a capital H on the screen as shown below. The vertical lines should be 100 pixels long and the horizontal line 50 pixels. (You can hide the turtle by adding the “Hide Turtle” command to the end of your program.)



5. Create a program called *Bow Tie* that draws the following figure. The triangle sides are all length 90.



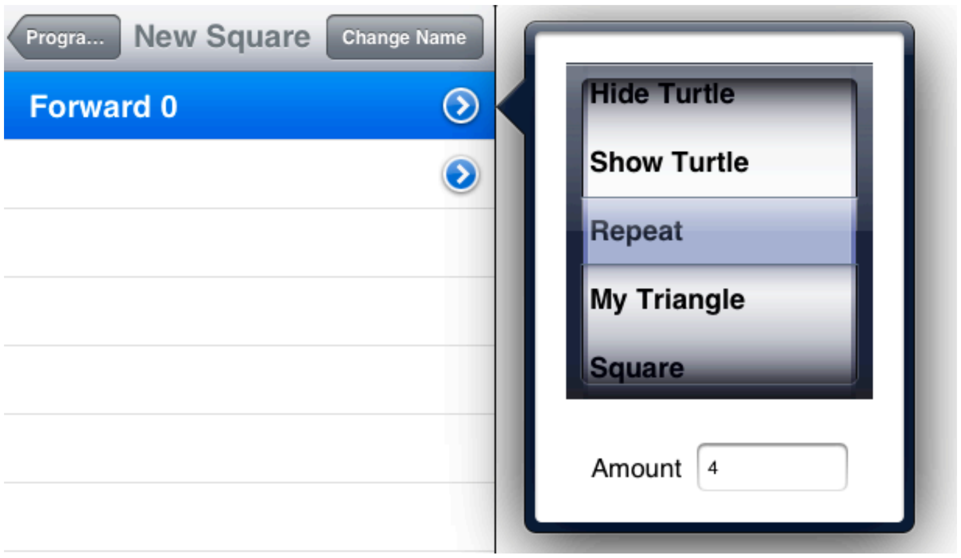
6. Create a new program called House. Create the following shape. All lines are length 90. Try reusing existing programs.



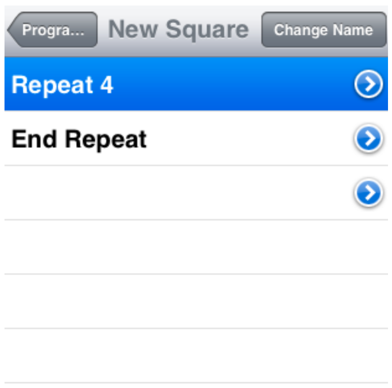
Repeat Command

The *Repeat* command executes a sequence of commands multiple times.

3. Create a new program called *New Square.*
   a. Add a new command
   b. Change the command name to *Repeat* and set the amount to 4, as shown below



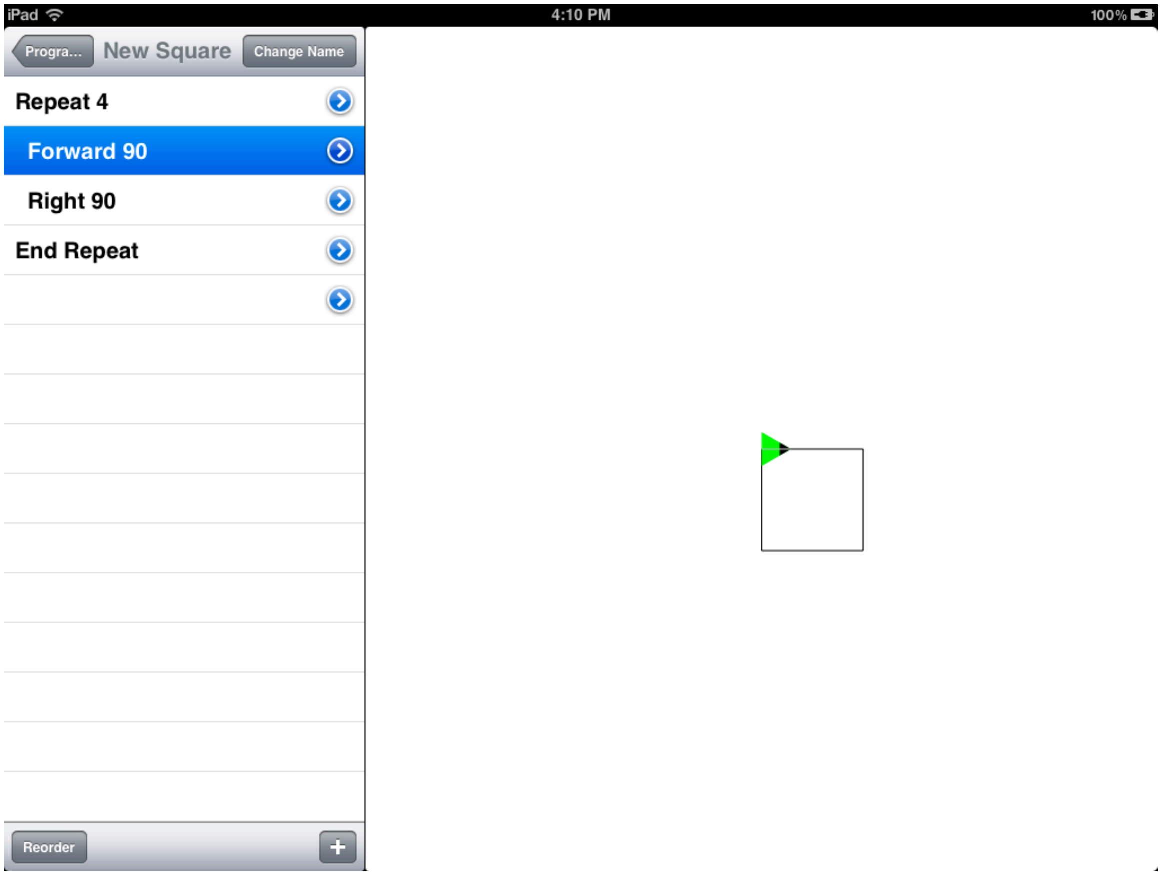
c. Touch outside the popup to update the command list, which should look like the following



d. Add a new command to the *Repeat* section by touching the button beside *End Repeat.*

c. Change the new command to *Forward 90.*
e. Add another new command to the *Repeat* section with *Right 90*.

You should end up with the following screen



Repeat 4
Forward 90
Right 90
End Repeat

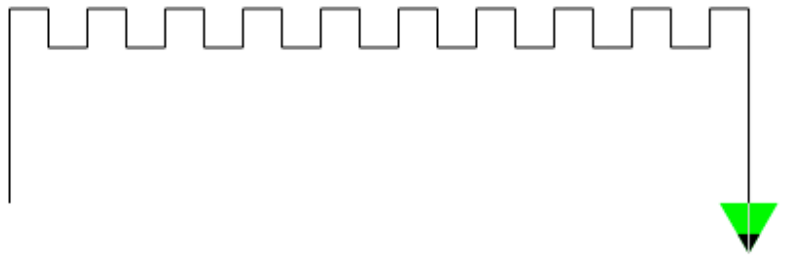
2. You can also include other programs you have written in a repeat statement. Create a new program, *Radioactive 2.* Add a *Repeat* statement with a count of 3. Add one command to the *Repeat* section, *My Triangle*. You should see the radioactive figure that you produced before.

3. Create a program called *Wall*.

a. Start the program with the following commands to move to the left

```
Right 180
Pen Up
Forward 200
Right 90
Pen Down
Forward 100
```

Now complete the program to draw the following image. There are 10 humps, each of height and width 20. The end edges are 100 pixels long.

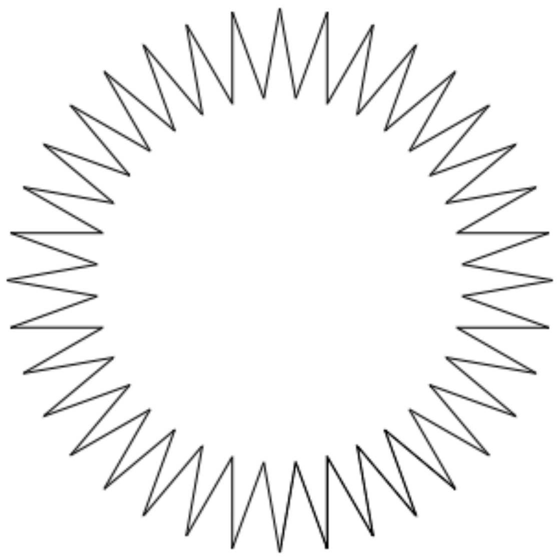


4. Make a program called *Circle*. There is no circle command, so we draw a circle as a sequence of short lines and small turns.

Draw a circle as a sequence of short line segments with the following program

```
Repeat 360
  Forward 3
  Right 1
End Repeat
```

5. Create a program called *Starburst.* Create the following figure. The lines are 50 pixels long. There are 36 outward points.



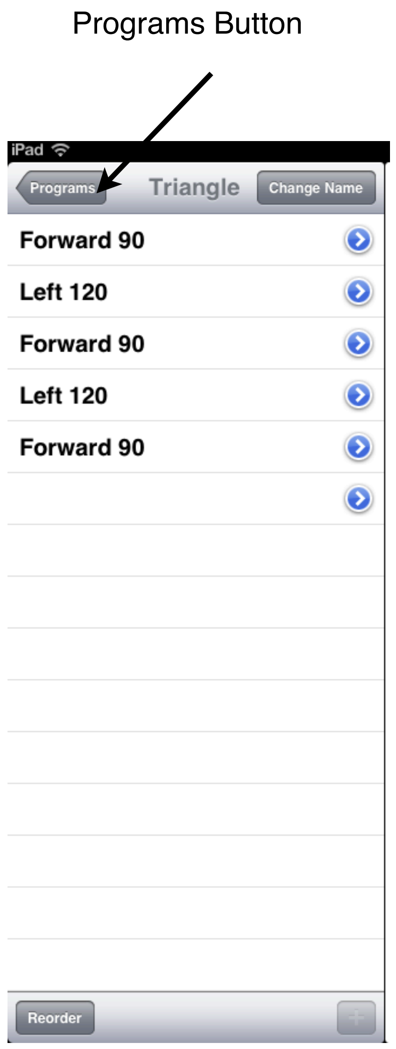
Appendix K

Logo Study - Low Cognitive Load Logo Instruction for Subprograms and Repeat Command

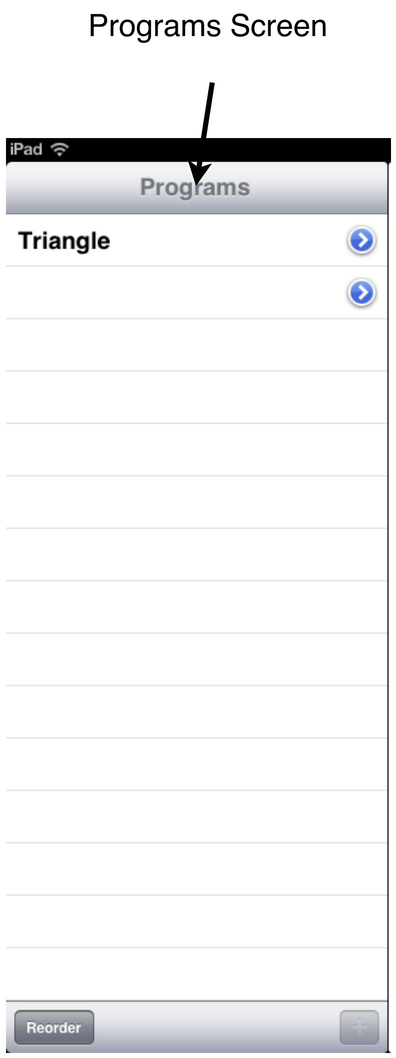
Subprograms

Do not use the Repeat command for tasks in this section.

1. If you are not at the programs screen, return to it by touching the programs button.

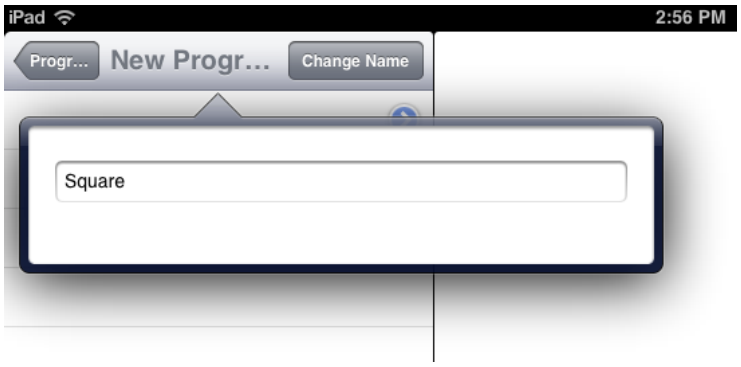


```
Forward 90
Left 120
Forward 90
Left 120
Forward 90
```



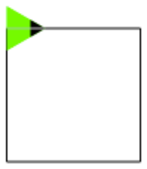
From the program listing screen, add a new program using the + button or using the › button beside the blank program.

Change the name of the new program to *Square* by using the Change Name button.

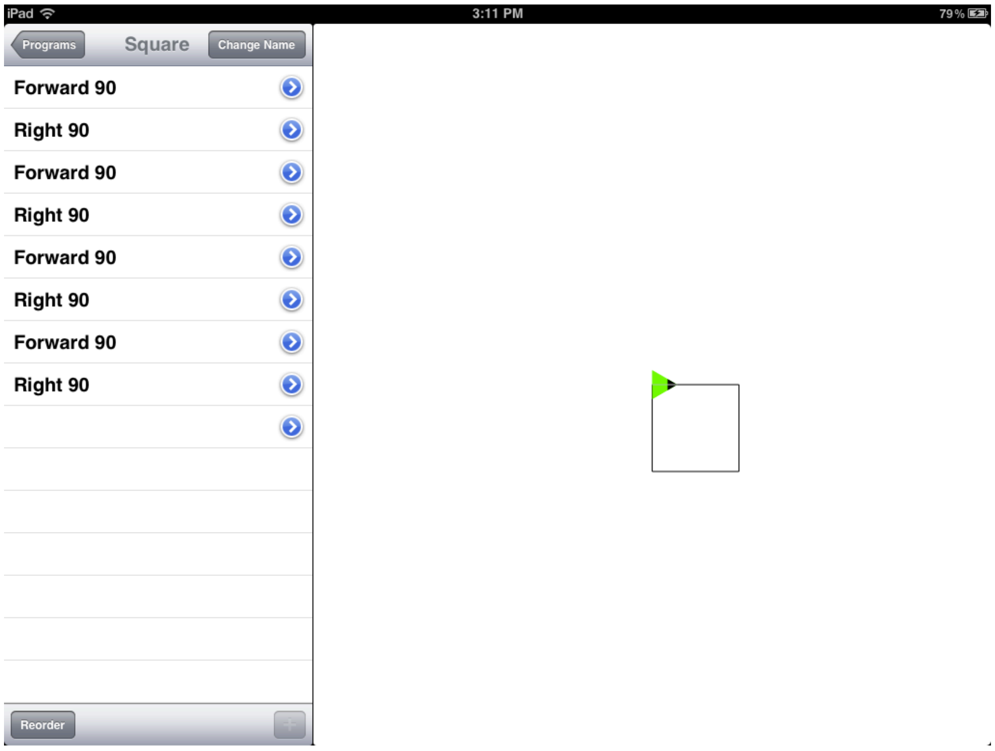


Square

We now want to draw the following image of a Square and leave the turtle in the position shown.



To draw the Square image, enter the commands shown below:



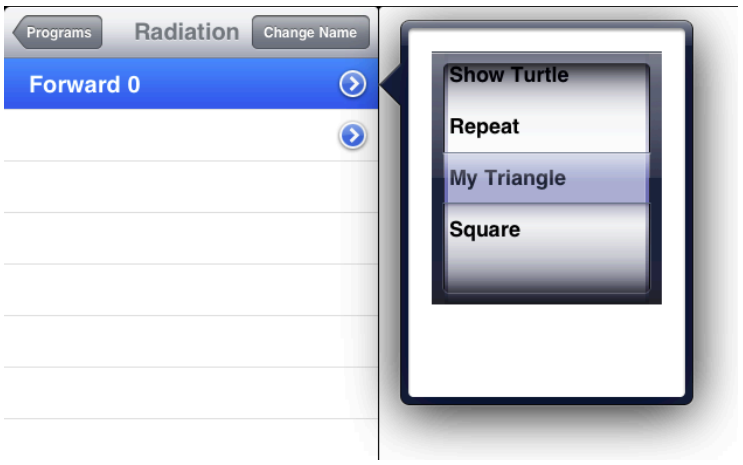
```
Forward 90
Right 90
Forward 90
Right 90
Forward 90
Right 90
Forward 90
Right 90
```

2. Create a new program called Pentagon. Add the following commands 5 times to draw the Pentagon.

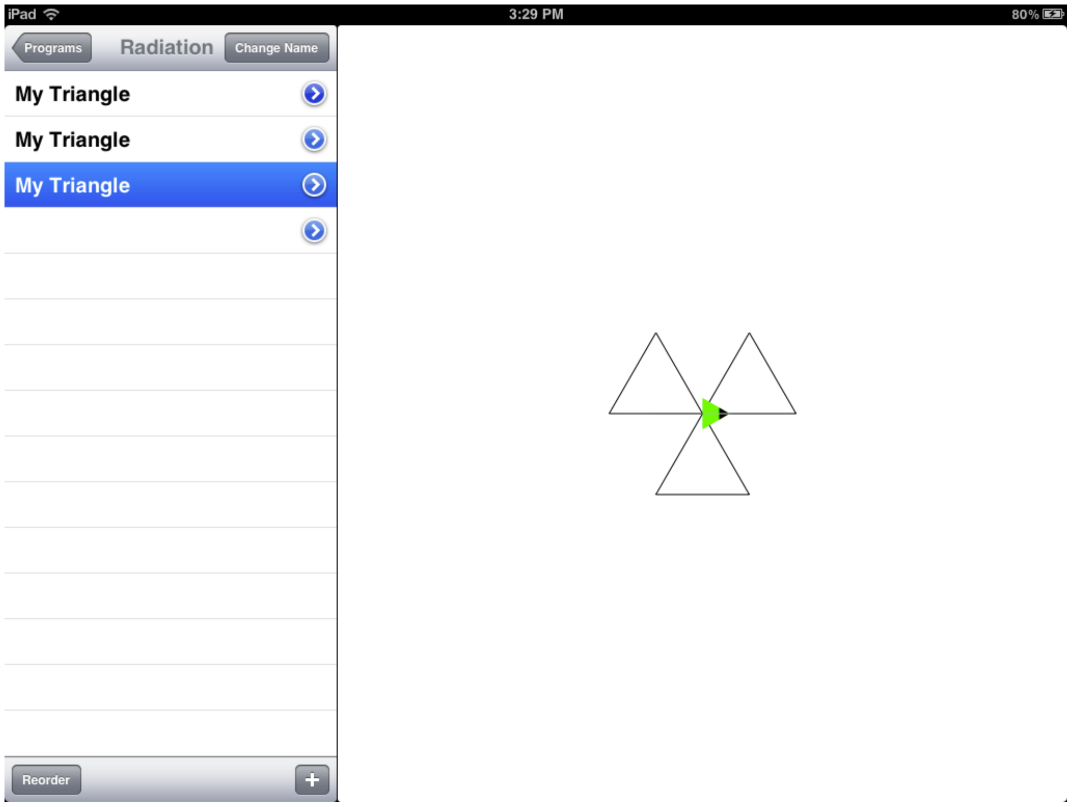
```
Forward 100
Right 72
```

3. You can also reuse existing programs. Return to the main program list and create a new program called *Radiation*.

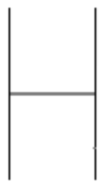
Add a new command and change the name of the command to *My Triangle*, as shown below.



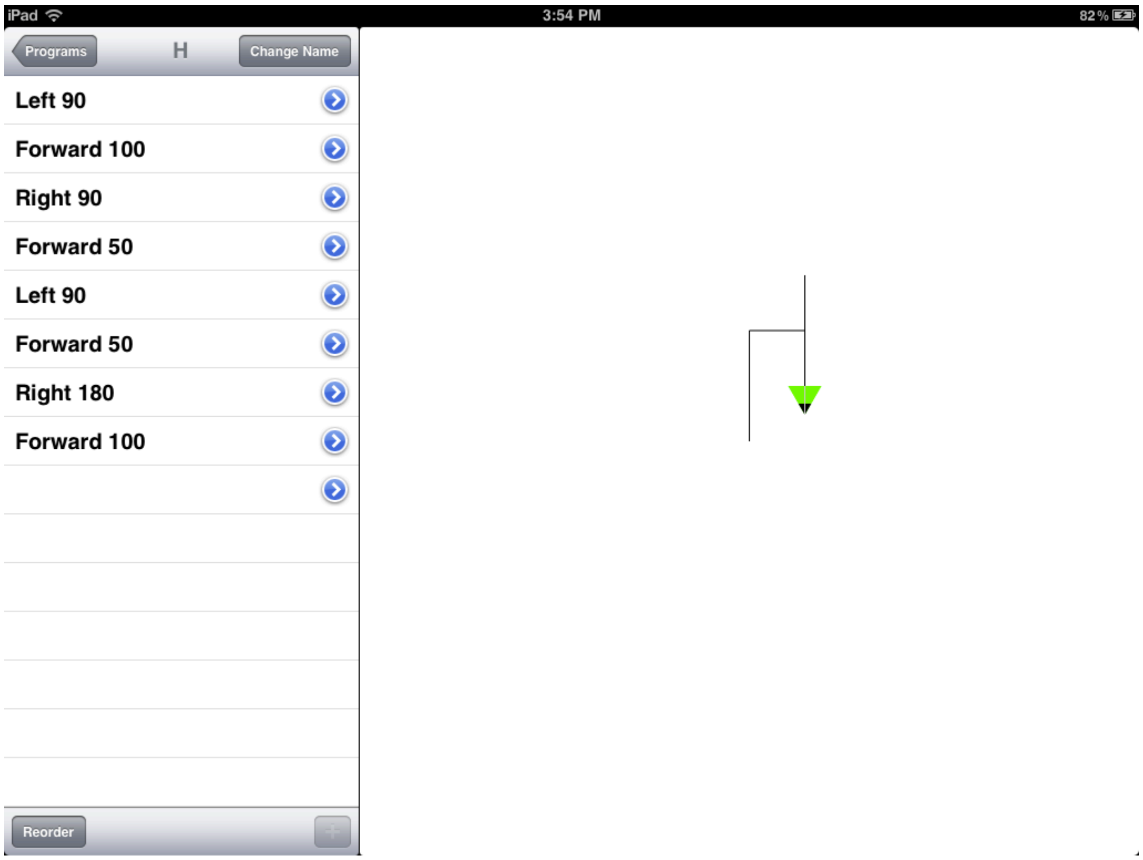
Repeat the above to add two more commands, both named *My Triangle,* to give the program shown below.



4. Create a new program named H. Setup the program to draw a capital H on the screen as shown below. The vertical lines should be 100 pixels long and the horizontal line 50 pixels. (You can hide the turtle by adding the "Hide Turtle" command to the end of your program.)

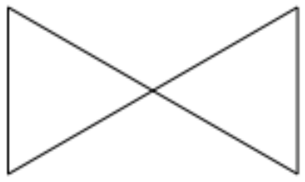


The following figure shows a nearly completed H program. You need to add a few commands to move the horizontal line to the centre of both vertical lines.

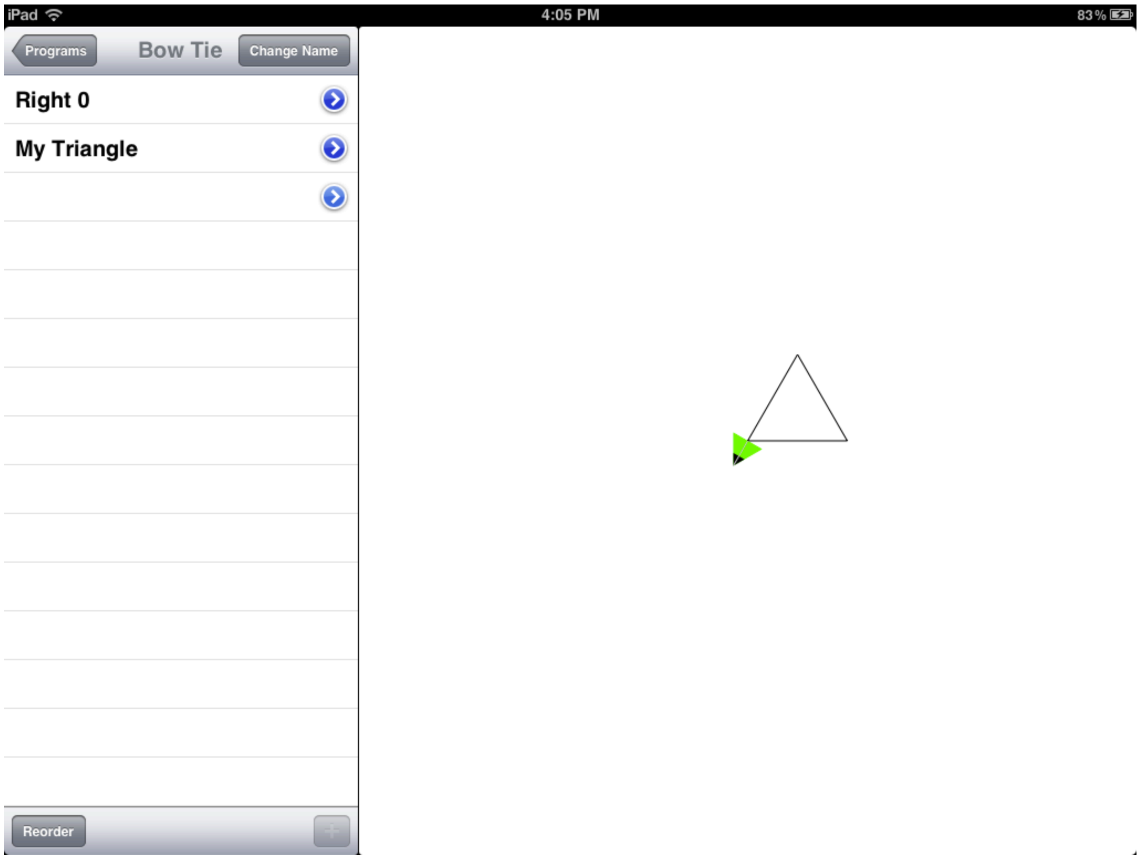


```
Left 90
Forward 100
Right 90
Forward 50
Left 90
Forward 50
Right 180
Forward 100
```

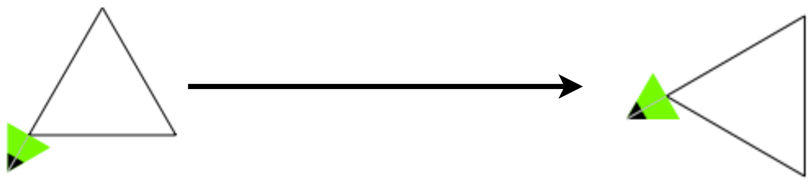
5. Create a program called *Bow Tie* that draws the following figure. The triangle sides are all length 90.



First add the following two lines to your program



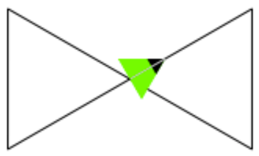
Now adjust the angle of the first command to give the right half of the Bow Tie.



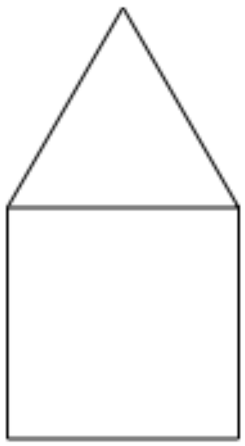
Now add another two commands to the end of the program:

```
        Right 0
        My Triangle
```

Again, adjust the angle of the new command to give the left side of the Bow Tie.



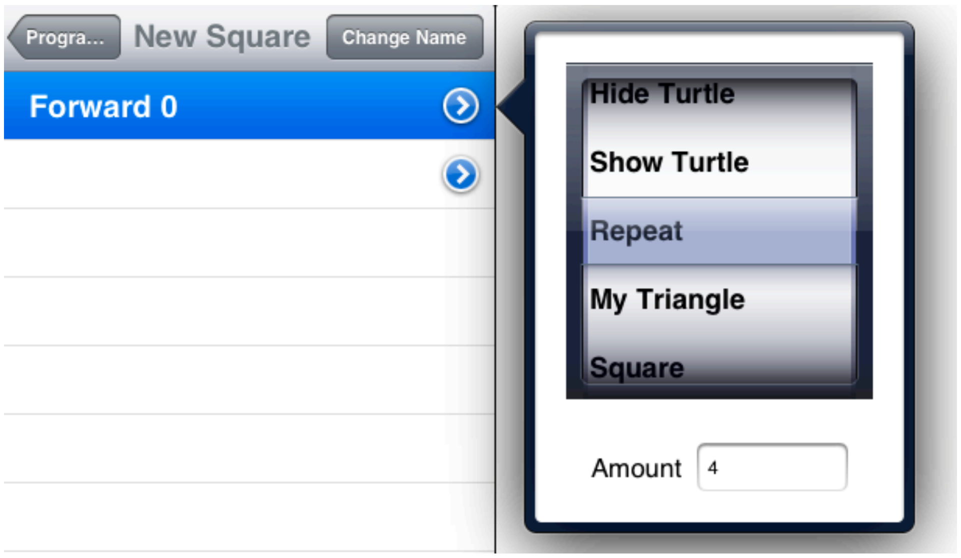
4. Create a new program called House. Create the following shape. All lines are length 90. Try reusing existing programs.



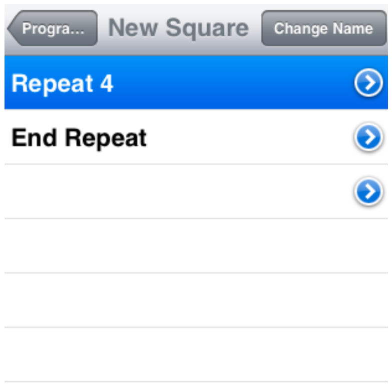
Repeat Command

The *Repeat* command executes a sequence of commands multiple times.

1. Create a new program called *New Square.*
a. Add a new command
b. Change the command to *Repeat* and set the amount to 4, as shown below



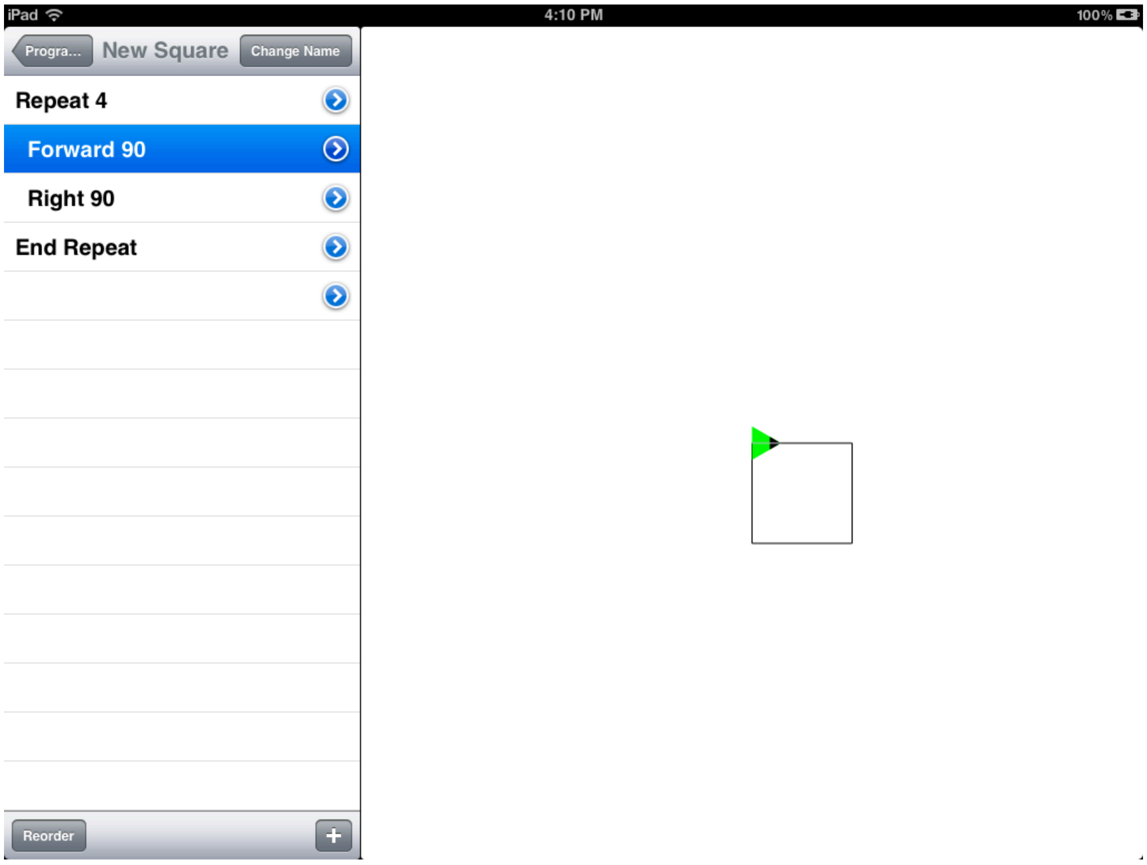
i. Touch outside the popup to update the list, which should look like the following



i. Add a new command to the *Repeat* section by touching the button beside *End Repeat.*

a. Change the new command to *Forward 90.*
b. Add another new command to the *Repeat* section with *Right 90*.

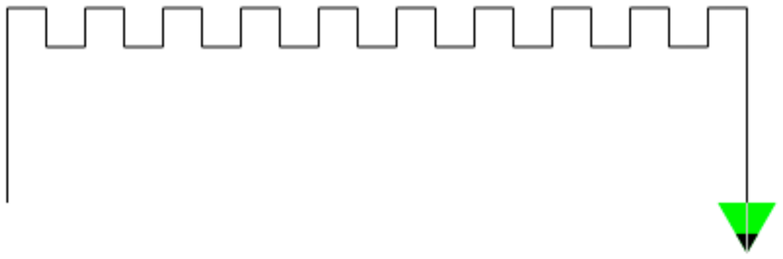
You should end up with the following screen



Repeat 4
Forward 90
Right 90
End Repeat

2. You can also include other programs you have written in a repeat statement. Create a new program, *Radioactive 2.* Add a *Repeat* statement with a count of 3. Add one command to the *Repeat* section, *My Triangle*. You should see the radioactive figure that you produced before.

3. Create a program called *Wall* that will display the following image



a. Start the program with the following commands to move left

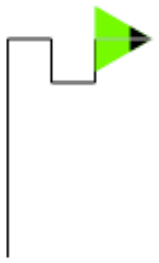
```
Right 180
Pen Up
Forward 200
Right 90
Pen Down
Forward 100
Right 90
```

Now we must draw the 10 humps in the Wall.

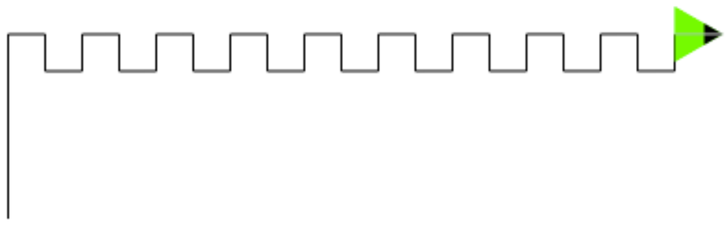
First we will draw one cycle in the hump.

Add a Repeat with count 1 to your program.

Add the commands inside the repeat to create the following image. The short lines are each 20 pixels long:



Now change the repeat count to 9. You should end up with an image that looks like the following:



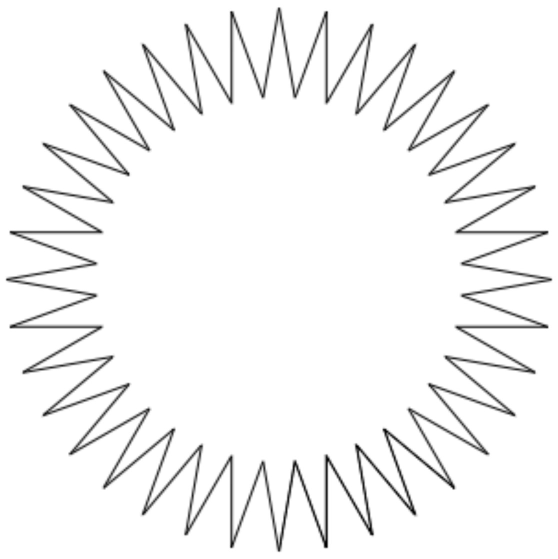
Add the top of the last Wall hump and then the 100 pixel end line.

4. Make a program called *Circle*. There is no circle command, so we draw a circle as a sequence of short lines and small turns.

Draw a circle as a sequence of short line segments with the following program

```
Repeat 360
  Forward 3
  Right 1
End Repeat
```

5. Create a program called *Starburst.* Create the following figure. The lines are 50 pixels long. There are 36 outward points.



Start by adding a Repeat command with a count of 1.

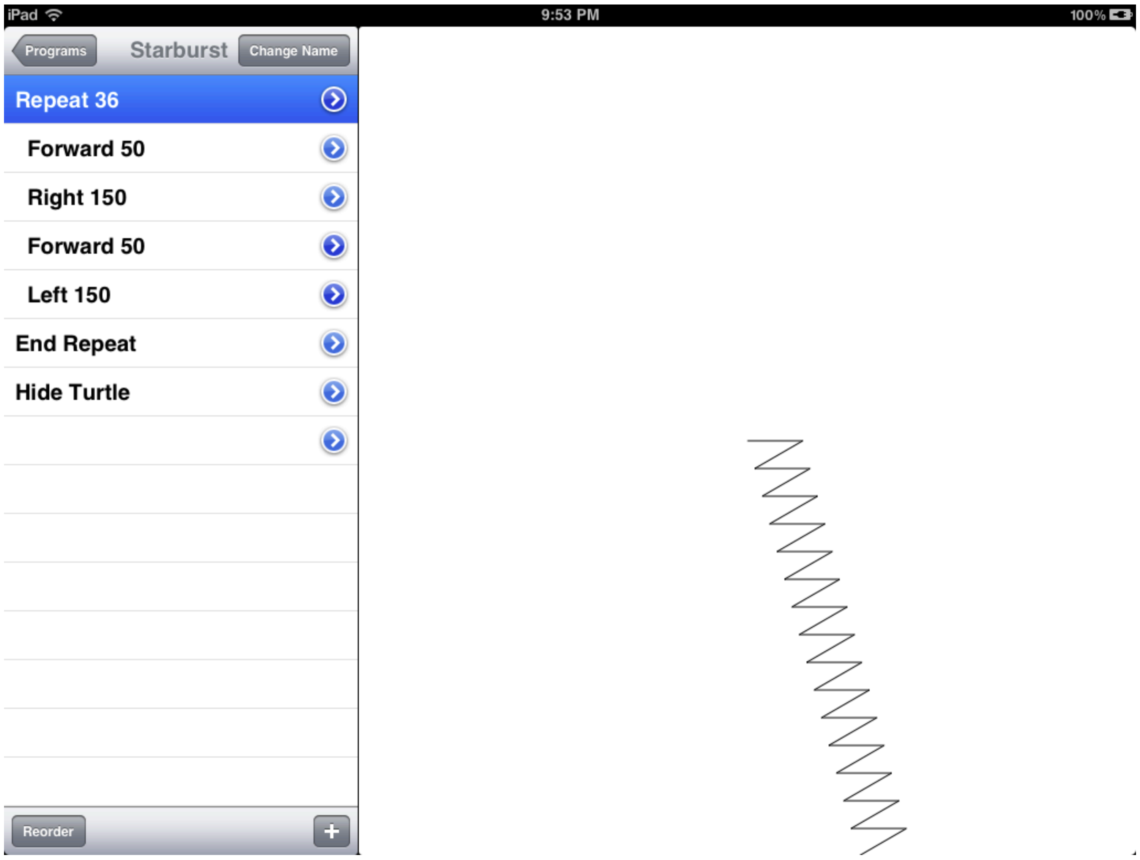
Add a Forward 50 command to draw the first line, which will go directly to the right.

Turn the turtle almost all the way back around with a *Right 150.*

Add a Forward 50 command to draw the second line. Then turn the turtle back around with a *Left 150.*

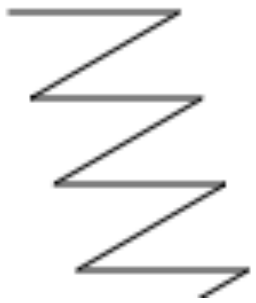
Now change the repeat to 36.

You should end up with the following.



```
Repeat 36
  Forward 50
  Right 150
  Forward 50
  Left 150
End Repeat
Hide Turtle
```

We need to make this zig-zag pattern curve into a circle. Looking more closely at the pattern, we see the following:



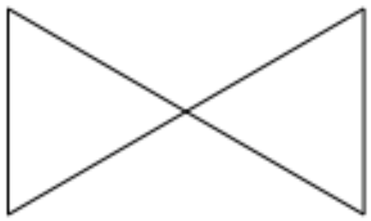
We see that there are two sets of parallel lines. To get the circular effect we will need to adjust the angles. Play with the angles on the *Right* and *Left* commands until you get the circular shape.

Appendix L

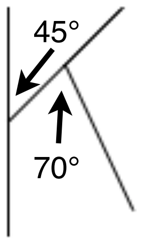
Logo Study - Subprograms Test

Do not use the repeat command.

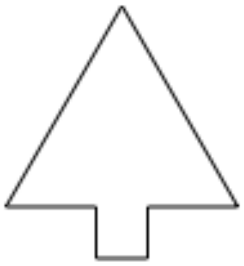
1. Create a program called *Bow Tie Two* that draws the following figure. The triangle sides are all length 90. Please do not look at your previous code or the instructional material.



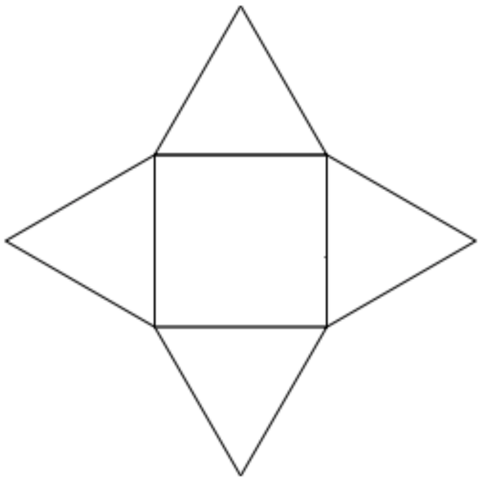
2. Create a new program named K. Setup the program to draw a capital K on the screen as shown below. The vertical lines should be 100 pixels long and the diagonal lines 70 pixels.



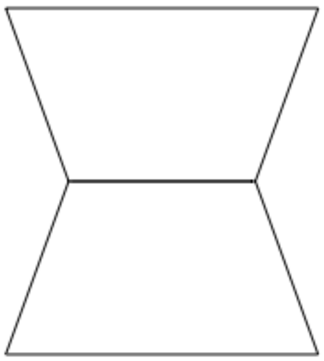
3. Create a program called *UpArrow*. Draw the following shape. All triangle sides are 90 pixels. All Square sides are 20 pixels.



4. Create a new program called *Star*. Draw the following image. All lines are length 90.



5. Draw the following image by creating one program (*Blade one*) to draw the top half and then a second program (*Blade two*) to draw the whole image. The short horizontal line and the diagonal lines are 100 pixels. The long horizontal lines are 170 pixels.

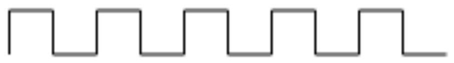


Appendix M

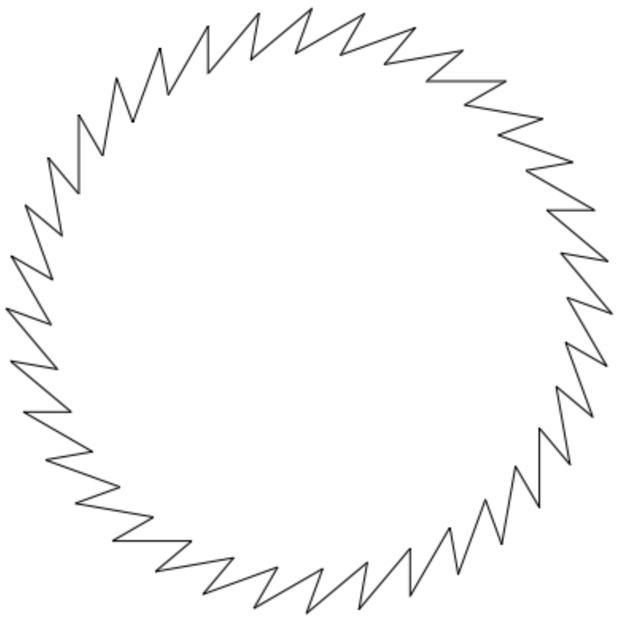
Logo Study - Repeat Test

1. Create a program called *Hexagon*. Create a six sided figure where each line is length 50 and each angle is a left turn of 60.

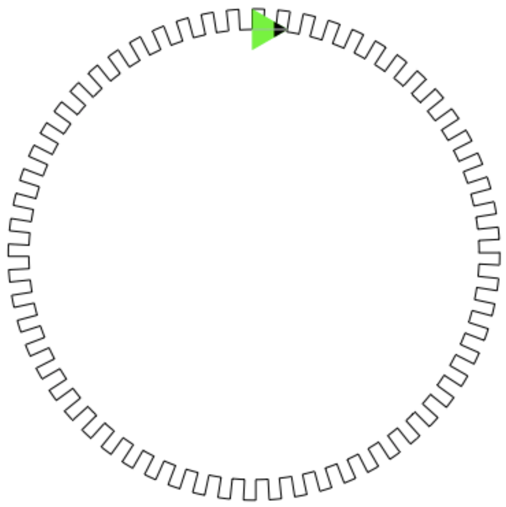
2. Create the following figure in a new program called *Bumps*. The lines are each 20 pixels long.



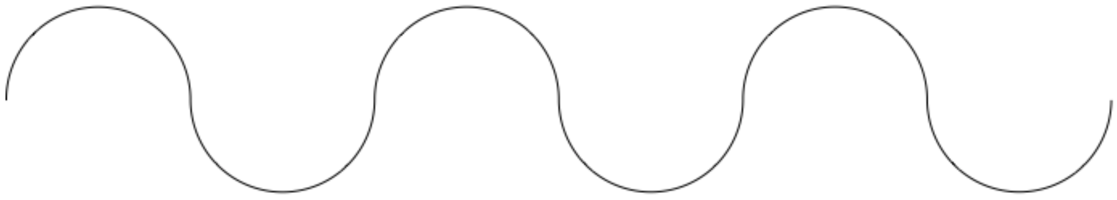
3. Create a program calls *Saw Blade.* Draw the following figure. There are 36 teeth. Long lines are 50 pixels long; short lines are 30 pixels long.



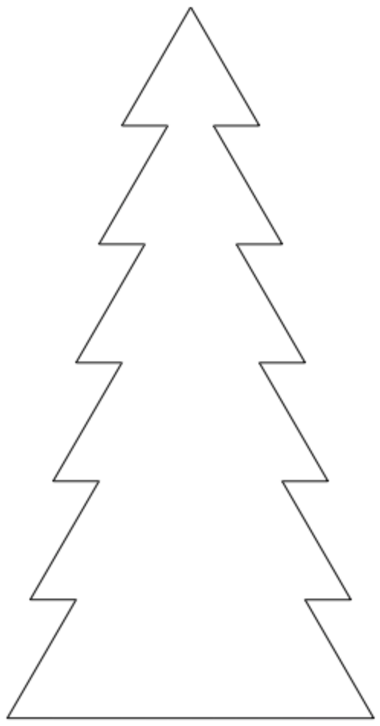
4. Create a program called *Gear*. Draw the following figure. There are 60 teeth. Each tooth is 9 pixels wide and 15 high. Each groove is 9 pixels wide.



5. Draw the following figure in a program called *Wave.* The height is up to you but 3 cycles should nearly fill the screen width (about 700 pixels wide).



6. Create a program called *Tree* and draw the following figure. The slanted lines are length 90 and the short horizontal lines are length 30.



Appendix N

Logo Study - Pre-Study Survey Results

The data in Table 97 shows the data from all participants for the survey. The Personal Experience column is a value between 0 and 3 indicating how highly the participant assessed their own programming experience. In the Logo Use and Mindstorms Use columns, a 1 indicates the participant has experience with Logo or Lego Mindstorms.

Table 97 - Logo participant survey results.

In the Logo Use and Mindstorms Use columns, a 1 indicates the participant has experience with Logo or Lego Mindstorms.

Participant	Condition	Personal Experience	Sex	Age	Program	Term	High school Courses	University Courses	Logo Use	Mindstorms Use
1	Low	3	M	19	SYDE	2A	2	1	0	1
2	High	1.5	F	19	Env Eng	2A	3	1	0	0
3	High	1	M	30	Civil Phd	17	0	1	0	0
4	High	2	M	19	CS	2B	2	4	0	1
5	Low	1	M	22	EE	4B	1	3	0	0
6	Low	1	M	23	Computing and Fin Mgmt	4B	3	8	0	0
7	High	0	M	22	BioChem	4B	0	0	0	0
8	Low	1	M	21	Masters Chem eng	5	2	1	0	0
9	Low	1	M	22	Civil	4B	1	1	0	0
10	Low	1	M	23	Arts and Business, honour psych	4B	2	4	0	1
11	High	1.5	M	18	CE	1B	4	3	0	1
12	Low	1	M	25	Masters ECE	2	2	2	0	0
13	Low	1	M	18	ECE	1B	0	2	0	1

Participant	Condition	Personal Experience	Sex	Age	Program	Term	High school Courses	University Courses	Logo Use	Mindstorms Use
14	High	1	M	20	biomedical science	3B	2	2	1	0
15	High	1	F	18	Chem eng	1B	0	1	0	0
16	High	1	F	18	Civil	1B	0	1	0	0
17	Low	2	M	19	CS	2B	0	5	0	0
18	Low	1	M	18	MecEng	3A	0	1	0	1
19	High	3	M	20	Knowledge Integration	4B	3	6	1	1
20	Low	1	M	22	Civil	4B	0	1	0	0
21	High	1.5	M	20	Actuarial Science	2B	3	2	0	0
22	High	0.5	M	21	Env and Business	4B	1	0	0	0
23	High	0	F	19	AHS	1B	2	0	0	0
24	Low	0	M	22	BioChem	3A	0	0	0	0
25	Low	0	F	25	Bio	4B	0	1	0	0
26	Low	0	F	28	Managment Science	Masters	0	0	0	0
27	High	0	F	24	English and poli sci	4B	0	0	0	0
28	High	1	F	19	Honours Arts	1B	0	2	0	0
29	Low	0	M	21	History	3	0	0	0	0
30	High	1	M	19	Arts and Business	1B	1	0	0	1
31	High	0	M	22	Psychology	4B	0	0	0	0
32	Low	0	F	23	Accounting	Masters	0	1	0	0
33	Low	0	M	22	Evn Bus	4B	0	0	0	0
34	High	0	M	22	biomedical science	4A	0	0	0	0
35	High	0.5	F	20	Legal Studies	3B	1	0	0	0
36	High	3	M	31	PhD Systems	5	2	20	0	0
37	High	0.5	F	18	Accounting	1B	1	0	0	0
38	Low	0	M	19	International Dev	1B	0	0	0	0

Participant	Condition	Personal Experience	Sex	Age	Program	Term	High school Courses	University Courses	Logo Use	Mindstorms Use
39	Low	0	M	19	Env and Business	1B	0	0	0	0
40	High	0	F	20	Env and resource studies	1B	0	0	0	0
41	High	0	M	18	Planning, BES	1B	0	0	0	0
42	Low	3	F	31	SYDE PhD	3	1	2	0	0
43	Low	3	M	24	SYDE Masters	6	1	4	0	0
44	High	0	F	22	Science	4B	1	0	0	0
45	High	0	F	20	International Dev	3B	0	0	0	0
46	Low	0	M	20	Honours Physc	3B	0	0	0	0
47	Low	1.5	M	25	Masters syde	4	3	1	0	0
48	Low	1	M	19	Chem eng	1B	0	0	0	0

Appendix O
Logo Study - Section Timings

Table 98 – Completion times in seconds for each participant for each section of the study.

Participant	Condition	Introduction Time	Subprograms Instruction Time	Subprograms Test Time	Repeat Instruction Time	Repeat Test Time
1	Low	891	1131	901	758	1326
2	High	597	807	1420	1333	2376
3	High	634	1686	2953	2582	2466
4	High	1176	700	1685	934	2486
5	Low	487	645	1415	650	1249
6	Low	574	651	1165	528	2458
7	High	558	644	1169	1780	1592
8	Low					
9	Low	1032	936	3051	1108	2746
10	Low	592	839	1296	846	2611
11	High	884	523	988	763	1444
12	Low	1115	1749	1671	1408	3929
13	Low	636	834	1606	777	1641
14	High	758	1341	1959	1791	3971
15	High	594	816	1448	2851	3292
16	High	647	908	2498	1392	1756
17	Low	509	648	1290	572	1162
18	Low	638	785	2250	808	1327
19	High	705	461	754	837	1297
20	Low	556	826	1426	774	1880

Participant	Condition	Introduction Time	Subprograms Instruction Time	Subprograms Test Time	Repeat Instruction Time	Repeat Test Time
21	High	644	775	1884	1749	2288
22	High	596	983	1604	649	1383
23	High	1053	1398	3592	2315	3639
24	Low	1145	819	2176	1292	3357
25	Low	705	1093	1708	1039	1844
26	Low	1022	812	2284	1027	2997
27	High	933	1199	2032	1267	1641
28	High	704	977	3624	2830	2519
29	Low	611	955	2263	834	2060
30	High	770	1436	2107	1396	3175
31	High	1191	959	2629	1349	2454
32	Low	482	581	1399	583	1365
33	Low	767	764	2879	1117	1581
34	High	736	856	1015	1225	2676
35	High	1123	1660	2418	905	2213
36	High	510	612	933	711	1421
37	High	704	935	1629	2843	2557
38	Low	555	713	1480	858	2136
39	Low	862	881	1691	694	2720
40	High	806	1010	1462	1096	1643
41	High	602	878	954	949	2326
42	Low	696	715	2064	892	2485
43	Low	442	698	1475	508	1179
44	High	1059	929	2091	1539	2447
45	High	1154	938	1803	3996	4366
46	Low	1410	1455	4439	1581	1096

Participant	Condition	Introduction Time	Subprograms Instruction Time	Subprograms Test Time	Repeat Instruction Time	Repeat Test Time
47	Low	788	920	1616	700	1992
48	Low	807	824	1739	956	3638

Appendix P

Logo Study - Problem Success and Timings for Subprogram Instructions

Table 99 – Success and time results for individual problems in the subprograms instruction section of Logo study.

A 1 in a success column indicates success at that problem. Times are in seconds.

		Square		Pentagon		Radiation		Bow Tie		House	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
1	Low	64	1	77	1	307	1	175	1	114	1
2	High	64	1	107	1	50	1	250	1	84	1
3	High	88	1	97	1	57	1	998	1	182	1
4	High	82	1	90	1	38	1	106	1	96	1
5	Low	108	1	76	1	35	1	138	1	40	1
6	Low	68	1	57	1	32	1	93	1	23	1
7	High	84	1	83	1	81	1	180	1	46	1
8	Low										
9	Low	114	1	74	1	53	1	257	1	244	1
10	Low	83	1	147	1	30	1	82	1	59	1
11	High	60	1	103	1	42	1	125	1	32	1
12	Low	102	1	215	1	208	1	279	1	51	1
13	Low	67	1	69	1	37	1	113	0	148	1
14	High	96	1	85	1	118	1	83	1	24	1
15	High	118	1	99	1	44	1	191	1	135	1
16	High	170	1	86	1	65	1	320	1	38	1
17	Low	75	1	76	1	45	1	56	1	43	1
18	Low	108	1	130	1	40	1	80	1	268	1

		Square		Pentagon		Radiation		Bow Tie		House	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
19	High	52	1	69	1	47	1	74	1	85	1
20	Low	76	1	73	1	35	1	241	1	101	1
21	High	62	1	143	1	42	1	187	1	27	1
22	High	101	1	91	1	46	1	106	1	63	1
23	High	85	1	87	1	190	1	312	1	49	1
24	Low	69	1	64	1	46	1	128	1	194	1
25	Low	107	1	85	1	88	1	152	0	152	1
26	Low	74	1	148	1	40	1	129	1	74	1
27	High	108	1	98	1	35	1	620	0	108	1
28	High	97	1	146	1	117	1	165	1	135	1
29	Low	85	1	69	1	46	1	322	1	84	1
30	High	118	1	108	1	53	1	547	0	292	1
31	High	93	1	100	1	37	1	284	0	112	1
32	Low	59	1	63	1	34	1	115	1	60	1
33	Low	85	1	88	1	58	1	135	1	105	1
34	High	76	1	88	1	46	1	275	1	170	1
35	High	140	1	202	1	588	1	288	1	36	1
36	High	95	1	69	1	24	1	155	1	58	1
37	High	63	1	86	1	42	1	240	1	211	1
38	Low	78	1	73	1	32	1	110	0	116	1
39	Low	68	1	80	1	44	1	99	1	69	1
40	High	114	1	107	1	249	1	133	1	226	1
41	High	71	1	77	1	42	1	475	1	34	1
42	Low	83	1	93	1	46	1	252	1	24	1
43	Low	58	1	83	1	36	1	123	1	35	1
44	High	119	1	96	1	61	1	240	1	153	1

		Square		Pentagon		Radiation		Bow Tie		House	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
45	High	121	1	130	1	68	1	191	1	170	1
46	Low	99	1	96	1	96	1	312	1	104	1

Appendix Q

Logo Study - Problem Success and Timings for Sub Programs Test Problems

Table 100 - Results for individual problems in the subprograms test section of Logo study.

A 1 in a success column indicated success at that problem. Times are in seconds.

Participant	Condition	Bow Tie Two Time	Bow Tie Two Success	Up Arrow Time	Up Arrow Success	Star Time	Star Success	Blade One Time	Blade One Success	Blade Two Time	Blade Two Success
1	Low	81	1	204	1	123	1	237	0		
2	High	158	1	154	1	304	1	434	1	135	1
3	High	267	1	242	1	516	1	704	1	697	0
4	High	250	1	305	1	251	1	121	1	330	1
5	Low	144	1	172	1	342	1	141	1	249	1
6	Low	172	1	196	1	215	1	272	1	124	1
7	High	84	1	240	1	180	1	284	1	152	1
8	Low										
9	Low	374	1	790	1	361	1	406	1	593	0
10	Low	126	1	172	1	254	1	364	1	100	1
11	High	81	1	245	1	208	1	153	1	68	1
12	Low	255	1	258	1	273	1	327	1	195	1
13	Low	208	0	238	0	143	0	168	0	557	0
14	High			448	1	709	1	495	1	37	1

Participant	Condition	Bow Tie Two Time	Bow Tie Two Success	Up Arrow Time	Up Arrow Success	Star Time	Star Success	Blade One Time	Blade One Success	Blade Two Time	Blade Two Success
15	High	210	1	156	1	375	1	224	1	226	1
16	High	203	1	314	1	304	1	1305	0		
17	Low	85	1	227	1	197	1	215	1	80	1
18	Low	134	1	224	1	268	1	645	1	658	1
19	High	60	1	107	1	166	1	174	1	61	1
20	Low	74	1	248	1	245	1	487	1	98	1
21	High	157	1	424	1	186	1	557	1	62	1
22	High	49	1	148	1	145	1	700	1	203	1
23	High			275	1	869	1	1613	1	403	0
24	Low	138	1	378	1	330	1	424	1	684	1
25	Low	280	0	436	1	327	1	223	1	182	1
26	Low	256	0	299	1	464	1	770	1	308	1
27	High	234	1	306	0	192	1	678	0	402	0
28	High	113	1	739	1	434	1	1610	0	230	0
29	Low	71	1	1119	1	190	1	362	1	282	1
30	High	259	1	197	1	332	1	223	1	489	1
31	High	191	0	427	1	274	1	766	0	393	0
32	Low	141	1	131	1	359	1	404	1	114	1
33	Low	268	1	738	1	234	1	520	1	852	1
34	High			208	1	293	0	183	1	198	1
35	High	210	0	227	1	491	1	468	1	716	1
36	High	117	1	154	1	157	1	242	1	39	1
37	High	141	1	280	1	326	1	212	1	232	1
38	Low	287	0	273	1	456	1	147	1	89	1
39	Low	55	1	732	0	337	1	212	1	127	1

Participant	Condition	Bow Tie Two Time	Bow Tie Two Success	Up Arrow Time	Up Arrow Success	Star Time	Star Success	Blade One Time	Blade One Success	Blade Two Time	Blade Two Success
40	High			175	1	392	1	233	1	435	1
41	High	64	1	121	1	233	1	238	1	127	1
42	Low	205	1	232	1	279	1	905	1	379	1
43	Low	55	1	164	1	171	1	477	1	419	1
44	High	241	1	250	1	257	1	588	1	295	1
45	High	123	1	300	1	428	1	538	1	150	1
46	Low	167	1	947	1	1104	1	1219	0	584	1
47	Low	129	1	287	1	214	1	498	1	230	1
48	Low	140	1	157	1	306	1	726	1	198	1

Appendix R

Logo Study - Problem Success and Timings for Repeat Instruction Problems

Table 101 - Results for individual problems in the Repeat Instruction section of Logo study. A 1 in a success column indicated success at that problem. Times are in seconds.

		New Square		Radioactive 2		Wall		Circle		Starburst	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
1	Low	64	1	30	1	301	1	54	1	201	1
2	High	67	1	36	1	593	1	32	1	506	1
3	High	73	1	41	1	635	1	36	1	1692	1
4	High	79	1	51	1	280	1	42	1	343	1
5	Low	64	1	35	1	223	1	32	1	227	1
6	Low	52	1	33	1	165	1	45	1	162	1
7	High	59	1	34	1	327	1	43	1	1244	1
8	Low										
9	Low	122	1	46	1	517	1	47	1	283	0
10	Low	71	1	31	1	314	1	45	1	315	1
11	High	57	1	26	1	276	1	43	1	205	1
12	Low	83	1	71	1	380	1	64	1	670	0
13	Low	93	1			304	0	82	1	208	0
14	High	90	1	33	1	374	1	43	1	1058	1
15	High	127	1	39	1	1923	1	36	1	605	1
16	High	64	1	90	1	520	1	27	1	596	0

		New Square		Radioactive 2		Wall		Circle		Starburst	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
17	Low	51	1	30	1	185	1	34	1	169	1
18	Low	48	1	161	1	286	1	29	1	213	1
19	High	46	1	59	1	211	1	28	1	444	1
20	Low	71	1	35	1	346	1	40	1	229	1
21	High	56	1	44	1	524	1	35	1	989	1
22	High	50	1	50	1	273	1	48	1	171	1
23	High	79	1	62	1	654	1	56	1	1308	0
24	Low	95	1	76	1	647	1	53	1	247	1
25	Low	65	1	44	1	380	1	33	1	362	0
26	Low	105	1	53	1	375	0	59	1	330	1
27	High	90	1	29	1	831	0	60	1	186	0
28	High	108	1	58	1	1772	0	132	1	620	0
29	Low	71	1	36	1	244	1	41	1	379	1
30	High	65	1	37	1	278	1	34	1	865	0
31	High	102	1	40	1	407	0	57	1	630	0
32	Low	68	1	37	1	179	1	33	1	222	1
33	Low	64	1	40	1	645	1	52	1	191	1
34	High	56	1	41	1	311	1	42	1	721	1
35	High	81	1	37	1	460	0	56	1	112	0
36	High	63	1	24	1	218	1	27	1	314	1
37	High	70	1	51	1	395	1	35	1	2197	1
38	Low	90	1	32	1	308	1	27	1	352	1
39	Low	89	1	32	1	156	1	32	1	313	1
40	High	103	1	38	1	470	0	46	1	336	0
41	High	67	1	34	1	203	1	42	1	551	1

		New Square		Radioactive 2		Wall		Circle		Starburst	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
42	Low	87	1	48	1	425	1	47	1	134	1
43	Low	68	1	30	1	222	1	33	1	92	1
44	High	69	1	54	1	734	1	42	1	478	1
45	High	117	1	42	1	1062	1	61	1	2614	1
46	Low	101	1	52	1	641	1	71	1	642	0
47	Low	82	1	58	1	258	1	55	1	164	1
48	Low	105	1	40	1	337	1	47	1	289	1

Appendix S

Logo Study - Problem Success and Timings for Repeat Test Problems

Table 102 - Results for individual problems in the repeat test section of Logo study.

A 1 in a success column indicated success at that problem. Times are in seconds.

		Hexagon		Bumps		Starburst		Gear		Wave		Tree	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
1	Low	58	1	85	1	201	1	147	1	244	0	608	1
2	High	118	1	140	1	506	1	277	1	442	1	539	1
3	High	166	1	126	1	1692	1	399	1	438	1	933	1
4	High	67	1	146	1	343	1	341	1	721	1	955	1
5	Low	40	1	149	1	227	1	147	1	273	1	355	1
6	Low	47	1	499	1	162	1	403	1	494	1	681	1
7	High	51	1	91	1	1244	1	204	1	340	1	558	1
8	Low												
9	Low	142	1	373	1	283	0	394	1	490	0	861	1
10	Low	34	1	110	1	315	1	311	1	391	1	1442	1
11	High	40	1	96	1	205	1	210	1	276	1	500	1
12	Low	94	1	119	1	670	0	361	1	892	0	1448	0
13	Low	107	1	129	0	208	0	258	0	268	0	594	0
14	High	91	1	265	1	1058	1	913	1	399	1	1182	1
15	High	57	1	155	1	605	1	255	1	536	1	2121	1
16	High	49	1	111	1	596	0	289	0	457	0	553	0

		Hexagon		Bumps		Starburst		Gear		Wave		Tree	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
17	Low	57	1			169	1	194	1	237	1	465	1
18	Low	50	1	111	1	213	1	239	1	174	1	451	1
19	High	39	1	93	1	444	1	110	1	164	1	514	1
20	Low	75	1	105	1	229	1	687	1	476	1	347	1
21	High	49	1	138	1	989	1	756	1	279	1	749	1
22	High	127	1	80	1	171	1	151	1	270	1	501	1
23	High	49	1	312	1	1308	0	534	1	694	1	1247	0
24	Low	96	1	124	1	247	1	861	1	382	1	823	1
25	Low	101	1	94	1	362	0	211	1	698	1	188	0
26	Low	74	1	112	1	330	1	173	1	1095	0	896	0
27	High	60	1	185	0	186	0	209	0	315	0	667	0
28	High	57	1	688	0	620	0	364	0	348	0	392	0
29	Low	45	1	75	1	379	1	179	1			717	1
30	High	89	1	194	1	865	0	220	1	1480	1	702	1
31	High	59	1	241	1	630	0	341	1	389	0	1117	0
32	Low	70	1	139	1	222	1	230	1	307	1	473	1
33	Low	77	1	116	1	191	1	388	0	370	0	449	0
34	High	82	1	195	1	721	1	302	0	602	0	424	1
35	High	113	1		0	112	0		0		0		0
36	High	40	1	93	1	314	1	183	1	226	1	434	1
37	High	46	1	120	1	2197	1	387	1	399	1	1154	1
38	Low	52	1	208	1	352	1	359	1	384	0	592	1
39	Low	40	1	102	1	313	1	239	1	1193	0	729	1
40	High	96	1	196	1	336	0	422	0	264	0	392	1
41	High	42	1	131	1	551	1	292	1	247	1	858	1

		Hexagon		Bumps		Starburst		Gear		Wave		Tree	
Participant	Condition	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success	Time	Success
42	Low	55	1	198	1	134	1	258	1	1000	1	530	1
43	Low	38	1	111	1	92	1	131	1	279	1	397	1
44	High	48	1	155	1	478	1	256	1	546	1	1268	1
45	High	65	1	167	1	2614	1	433	0	669	1	1385	1
46	Low	64	1	485	1	642	0	153	0		0		0
47	Low	59	1	82	1	164	1	131	1	416	1	730	1
48	Low	119	1	204	1	289	1	382	1	722	1	1145	1

Appendix T
Logo Study - Bow Tie and Bow Tie Two Subprogram Use

Table 103 shows the times, success, and use of subprograms for the Bow Tie and Bow Tie Two programs. A 1 in the subprograms columns indicates the My Triangle program was used in the Bow Tie or Bow Tie Two program.

Table 103 – Bow Tie and Bow Tie Two subprogram use

		Bow Tie			Bow Tie Two		
Participant	Condition	Time	Success	Subprogram	Time	Success	Subprogram
1	Low	175	1	1	81	1	1
2	High	250	1	1	158	1	0
3	High	998	1	1	267	1	0
4	High	106	1	0	250	1	0
5	Low	138	1	1	144	1	1
6	Low	93	1	1	172	1	0
7	High	180	1	0	84	1	0
8	Low						
9	Low	257	1	1	374	1	1
10	Low	82	1	1	126	1	1
11	High	125	1	1	81	1	1
12	Low	279	1	1	255	1	1
13	Low	113	0	1	208	0	1
14	High	83	1	1	9		2
15	High	191	1	1	210	1	0
16	High	320	1	1	203	1	0
17	Low	56	1	1	85	1	1
18	Low	80	1	1	134	1	0
19	High	74	1	1	60	1	1
20	Low	241	1	1	74	1	1
21	High	187	1	1	157	1	1
22	High	106	1	1	49	1	0
23	High	312	1	0	41		2
24	Low	128	1	1	138	1	1
25	Low	152	0	1	280	0	1
26	Low	129	1	1	256	0	0
27	High	620	0	0	234	1	0

		Bow Tie			Bow Tie Two		
Participant	Condition	Time	Success	Subprogram	Time	Success	Subprogram
28	High	165	1	0	113	1	0
29	Low	322	1	1	71	1	1
30	High	547	0	0	259	1	1
31	High	284	0	0	191	0	0
32	Low	115	1	1	141	1	0
33	Low	135	1	1	268	1	1
34	High	275	1	0			
35	High	288	1	0	210	0	1
36	High	155	1	0	117	1	0
37	High	240	1	0	141	1	0
38	Low	110	0	1	287	0	1
39	Low	99	1	1	55	1	1
40	High	133	1	0			
41	High	475	1	0	64	1	0
42	Low	252	1	1	205	1	0
43	Low	123	1	1	55	1	1
44	High	240	1	0	241	1	1
45	High	191	1	0	123	1	0
46	Low	312	1	1	167	1	1
47	Low	344	1	1	129	1	1
48	Low	119	1	1	140	1	0

Appendix U
Logo Study – Subprogram Use in House Problem

Table 104 – Use of Triangle and Square subprograms in House problem.

1 indicates use of the subprogram.

Participant	Condition	Triangle Subprogram	Square Subprogram
1	Low	1	0
2	High	1	1
3	High	0	0
4	High	1	1
5	Low	1	1
6	Low	1	1
7	High	1	1
8	Low		
9	Low	1	1
10	Low	1	1
11	High	1	1
12	Low	1	1
13	Low	1	0
14	High	1	1
15	High	1	0
16	High	1	1
17	Low	1	1
18	Low	1	1
19	High	1	1
20	Low	1	1
21	High	1	1
22	High	1	0
23	High	1	1
24	Low	1	1
25	Low	1	1
26	Low	1	1
27	High	1	1
28	High	1	1
29	Low	1	1
30	High	1	1
31	High	1	0
32	Low	1	1

Participant	Condition	Triangle Subprogram	Square Subprogram
33	Low	1	1
34	High	1	1
35	High	1	1
36	High	1	1
37	High	1	1
38	Low	1	0
39	Low	1	0
40	High	0	0
41	High	1	1
42	Low	1	1
43	Low	1	1
44	High	1	1
45	High	1	0
46	Low	1	1
47	Low	1	1
48	Low	1	1

Appendix V
Logo Study – Subprogram use in Star Problem

Table 105 - Use of Triangle, Square, and House subprograms in House problem.

The number of times each subprogram was used is indicated in each cell.

Participant	Condition	Triangle Subprogram	Square Subprogram	House Subprogram
1	Low	4	0	0
2	High	0	0	0
3	High	0	0	0
4	High	4	1	0
5	Low	4	1	0
6	Low	4	0	0
7	High	4	1	0
8	Low			0
9	Low	4	0	0
10	Low	4	0	0
11	High	4	1	0
12	Low	0	0	0
13	Low	1	0	0
14	High	4	1	0
15	High	4	0	0
16	High	1	1	0
17	Low	4	0	0
18	Low	0	0	0
19	High	4	1	0
20	Low	4	0	0
21	High	4	1	0
22	High	4	0	0
23	High	0	0	0
24	Low	4	1	0
25	Low	4	1	0
26	Low	4	1	0
27	High	4	1	0
28	High	4	1	0
29	Low	4	1	0
30	High	0	1	0
31	High	0	0	0
32	Low	4	1	0

Participant	Condition	Triangle Subprogram	Square Subprogram	House Subprogram
33	Low	4	1	0
34	High	0	0	4
35	High	2	1	0
36	High	4	0	0
37	High	1	1	1
38	Low	1	0	0
39	Low	1	1	1
40	High	0	0	0
41	High	4	1	0
42	Low	4	0	0
43	Low	4	0	0
44	High	4	0	0
45	High	1	1	0
46	Low	2	1	0
47	Low	4	0	0
48	Low	0	0	0

Appendix W
Logo Study – Subprogram use in Bow Tie and Bow Tie Two

Table 106 - Use of Triangle subprograms in Bow Tie and Bow Tie Two problems.

1 indicates use of the subprogram.

Participant	Condition	Bow Tie Triangle Subprogram	Bow Tie Two Triangle Subprogram
1	Low	1	1
2	High	1	0
3	High	1	0
4	High	0	0
5	Low	1	1
6	Low	1	0
7	High	0	0
8	Low		
9	Low	1	1
10	Low	1	1
11	High	1	1
12	Low	1	1
13	Low	1	1
14	High	1	2
15	High	1	0
16	High	1	0
17	Low	1	1
18	Low	1	0
19	High	1	1
20	Low	1	1
21	High	1	1
22	High	1	0
23	High	0	2
24	Low	1	1
25	Low	1	1
26	Low	1	0
27	High	0	0
28	High	0	0
29	Low	1	1

Participant	Condition	Bow Tie Triangle Subprogram	Bow Tie Two Triangle Subprogram
30	High	0	1
31	High	0	0
32	Low	1	0
33	Low	1	1
34	High	0	
35	High	0	1
36	High	0	0
37	High	0	0
38	Low	1	1
39	Low	1	1
40	High	0	
41	High	0	0
42	Low	1	0
43	Low	1	1
44	High	0	1
45	High	0	0
46	Low	1	1
47	Low	1	1
48	Low	1	0

Appendix X

Logo Study – Use of Blade One in Blade Two

Table 107 – Number of times each participants used Blade One as a Subprogram in Blade Two.

Participant	Condition	Blade Two Subprograms
1	Low	0
2	High	2
3	High	0
4	High	2
5	Low	2
6	Low	2
7	High	2
8	Low	0
9	Low	1
10	Low	0
11	High	2
12	Low	2
13	Low	0
14	High	0
15	High	2
16	High	0
17	Low	2
18	Low	2
19	High	2
20	Low	2
21	High	2
22	High	2
23	High	1
24	Low	2
25	Low	2
26	Low	2
27	High	1
28	High	2
29	Low	1
30	High	2

Participant	Condition	Blade Two Subprograms
31	High	1
32	Low	2
33	Low	0
34	High	2
35	High	1
36	High	2
37	High	1
38	Low	2
39	Low	1
40	High	0
41	High	2
42	Low	2
43	Low	2
44	High	2
45	High	2
46	Low	1
47	Low	2
48	Low	2

Appendix Y
Logo Study – Loop Use in Hexagon and Tree

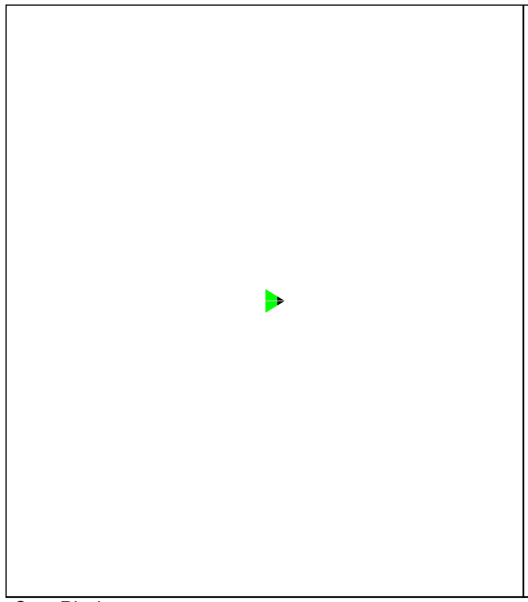
Table 108 – Number of times each participant used a repeat command in solution to Hexagon and Tree.

Participant	Condition	Hexagon Repeats	Tree Repeats
1	Low	1	2
2	High	1	2
3	High	1	2
4	High	1	2
5	Low	1	2
6	Low	1	2
7	High	1	2
8	Low		
9	Low	1	3
10	Low	1	2
11	High	1	2
12	Low	1	2
13	Low	0	0
14	High	1	2
15	High	1	2
16	High	1	2
17	Low	1	2
18	Low	1	2
19	High	1	2
20	Low	1	2
21	High	1	2
22	High	1	2
23	High	1	2
24	Low	1	2
25	Low	1	0
26	Low	1	2
27	High	1	0
28	High	1	1
29	Low	1	2
30	High	1	2
31	High	1	0
32	Low	1	2

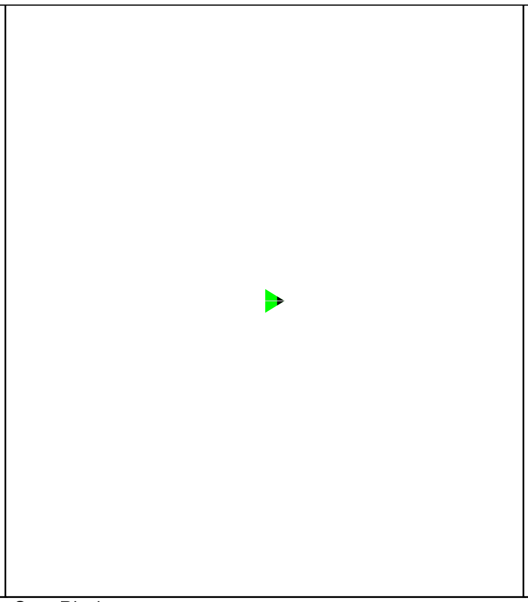
Participant	Condition	Hexagon Repeats	Tree Repeats
33	Low	1	2
34	High	1	2
35	High	1	
36	High	1	2
37	High	1	2
38	Low	1	2
39	Low	1	1
40	High	0	0
41	High	1	2
42	Low	1	2
43	Low	1	2
44	High	1	2
45	High	1	1
46	Low	1	
47	Low	1	2
48	Low	1	2

Appendix Z

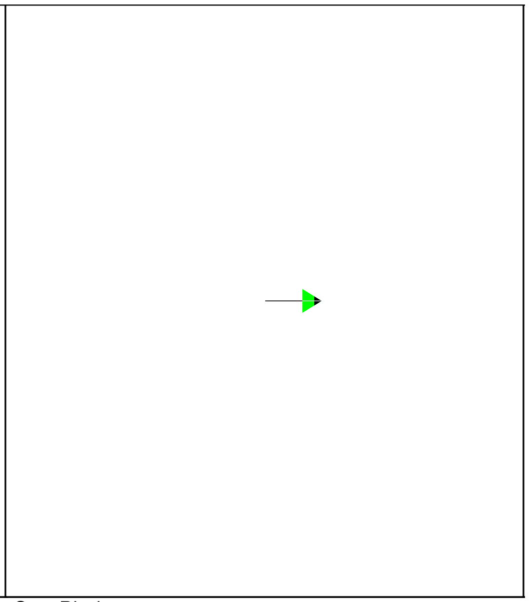
Logo Study - Saw Blade Solution Steps for Participant 32



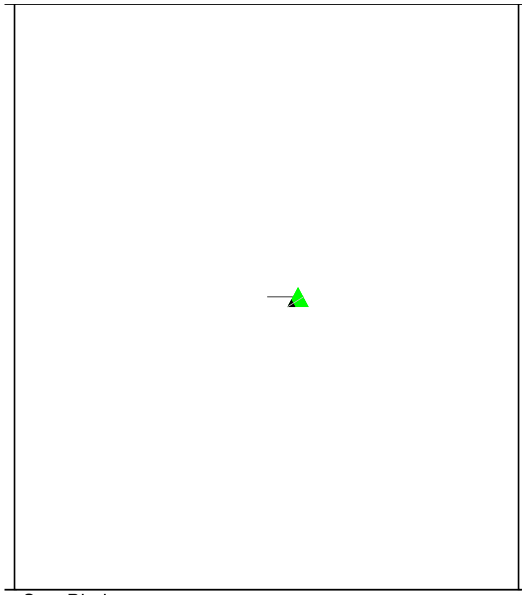
```
Saw Blade
  Repeat 0
  End Repeat
```



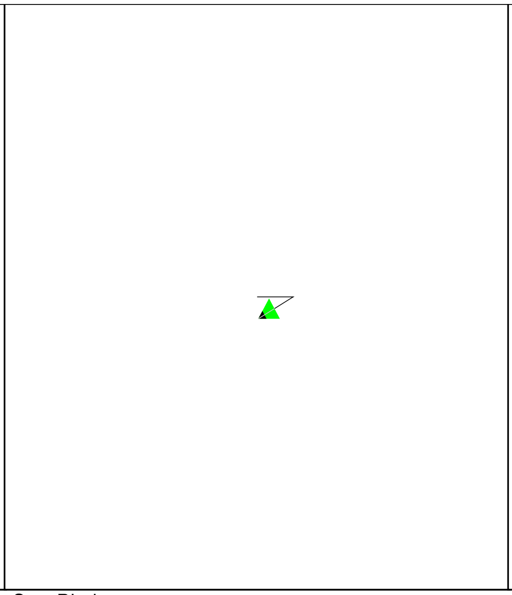
```
Saw Blade
  Repeat 1
  End Repeat
```



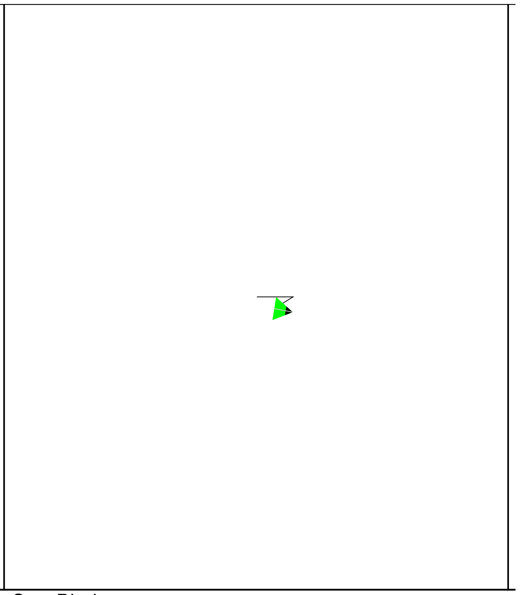
```
Saw Blade
  Repeat 1
    Forward 50
  End Repeat
```



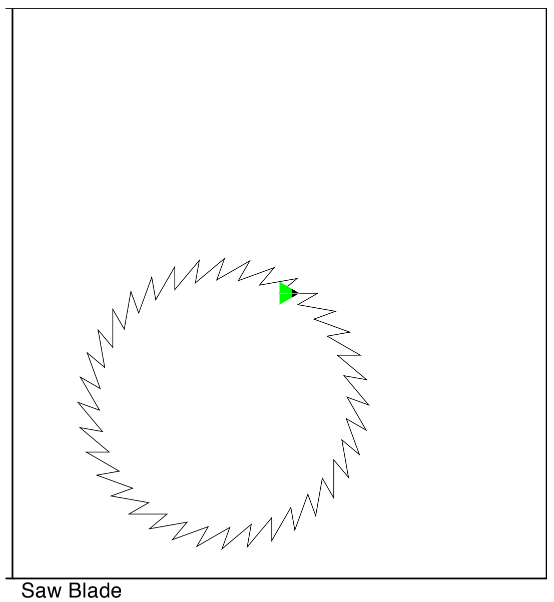
```
Saw Blade
  Repeat 1
    Forward 50
    Right 150
  End Repeat
```



```
Saw Blade
  Repeat 1
    Forward 50
    Right 150
    Forward 30
  End Repeat
```



```
Saw Blade
  Repeat 1
    Forward 50
    Right 150
    Forward 30
    Left 140
  End Repeat
```



```
Saw Blade
  Repeat 36
    Forward 50
    Right 150
    Forward 30
    Left 140
  End Repeat
```

Appendix AA

Logo Study - Zig-Zag step use and angle adjustments for Starburst and Saw Blade problems

Table 109 – Number of Angle adjustments and range of angles used for Starburst and Saw Blades problems.

		Starburst			Sawblade		
Participant	Condition	Number of Adjustments	Min	Max	Number of Adjustments	Min	Max
1	Low	12	149	180	2	150	160
2	High	13	0	180	15	10	170
3	High	46	1	200	21	1	135
4	High	1	90	180	0	90	170
5	Low	4	10	350	0	30	155
6	Low	1	150	160	6	140	170
7	High	30	0	370	2	15	330
8	Low						
9	Low	18	75	150	23	50	160
10	Low	6	90	180	2	120	150
11	High	6	150	165	9	90	165
12	Low	36	1	195	51	15	360
13	Low	17	10	180			

		Starburst			Sawblade		
Participant	Condition	Number of Adjustments	Min	Max	Number of Adjustments	Min	Max
14	High	35	0	324	3	10	170
15	High	22	4	310	0	150	200
16	High	4	30	140	1	65	100
17	Low	5	145	160	1	150	160
18	Low	11	50	180	4	120	170
19	High	6	1	200	8	1	190
20	Low	13	140	180	3	145	160
21	High	37	0	350	5	145	165
22	High	10	150	180	9	60	180
23	High	19	0	350	11	10	170
24	Low	3	150	160	18	0	180
25	Low	23	30	270	21	1	450
26	Low	1	0	160	24	15	360
27	High	2	25	155	1	130	130
28	High	31	0	360	52	1	180
29	Low	39	0	180	4	150	170
30	High	10	1	160	4	3	160
31	High	14	1	200	15	40	190
32	Low	15	30	180	0	140	150
33	Low	9	90	160	1	150	160
34	High	27	1	240	14	28	180
35	High	2	30	45			

		Starburst			Sawblade		
Participant	Condition	Number of Adjustments	Min	Max	Number of Adjustments	Min	Max
36	High	7	1	180	15	90	180
37	High	63	0	180	8	0	140
38	Low	33	10	300	6	100	170
39	Low	29	0	360	36	10	180
40	High	2	1	90	2	45	360
41	High	16	10	359	38	10	1100
42	Low	1	150	160	3	45	145
43	Low	1	140	150	3	130	150
44	High	22	0	190	0	140	150
45	High	104	0	360	50	35	315
46	Low	66	5	500	0	150	150
47	Low	1	10	150	4	10	150
48	Low	9	150	180	6	145	160

Appendix BB
Logo Study - NASA-TLX Results

Table 110 - NASA-TLX results for Logo Study.

Participant	Condition	Introduction	Subprogram Instruction	Subprogram Test	Repeat Instruction	Repeat Test
1	Low		5.3	2.98	1.79	5.82
2	High		6.49	7.42	7.62	7.31
3	High	6.61	6.11	6.92	6.72	5.3
4	High	2.47	3.97	5.55	3.45	7.3
5	Low	3.64	3.94	6.2	5.86	6.99
6	Low	0	2.03	3.27	4.1	5.41
7	High	2.1	1.93	2.08	3.12	1.79
8	Low	6.42	7.18	1.52	5.16	3.15
9	Low	6.64	7.08	8.45	7.13	8.05
10	Low	4.41	4.64	6	6.25	7.01
11	High	2.39	3.07	4.37	4.35	5.24
12	Low	4.7	5.28	6.74	6.43	7.78
13	Low	4.72	5.23	4.83	5.76	6.61
14	High	0.39	1.1	1.69	1.48	2.99
15	High	2.5	4.88	5.9	7.29	7.29
16	High	3.57	5.51	8.65	7.2	8.76
17	Low	3.94	4.13	3.64	2.55	3.98
18	Low	2.6	3.87	6.16	4.24	6.5
19	High	2.36	1.8	1.8	2.3	3.4
20	Low	3.95	4.17	4.52	4.86	6.81
21	High	2.26	3	3.52	4.4	5.24
22	High	4.24	3.44	3.15	3.81	3.24
23	High	3.78	3.98	6.83	7.75	7.55
24	Low	1.28	3.4	5.64	4.26	5.53
25	Low	5.82	5.93	6.23	6.4	7.77
26	Low	4.97	6.03	6.44	2.68	6.98
27	High	3.05	4.38	7.05	8.11	8
28	High	3.2	3.14	4.27	2.92	3.37
29	Low	0.75	2.17	5.4	2.38	3.6
30	High	5.4	7.53	6.57	8.56	8.94
31	High	3.61	5.21	7.99	7.88	8.88
32	Low	0.07	0.27	1.24	0.37	1.75

33	Low	4.83	5.22	7.59	8.48	8.62
34	High	5.6	5.27	5.28	4.57	6.24
35	High	0.16	1.35	0.79	1.02	
36	High	3.17	1.72	3.16	3.03	2.72
37	High	3.03	4.18	4.47	5.83	4.62
38	Low	0.14	0.77	1.02	1.2	1.21
39	Low	5.18	6.27	7.77	6.16	6.01
40	High	5.27	4.1	6.06	5.45	5.66
41	High	1.21	2.85	3.61	5.53	7.01
42	Low	5.43	1.06	1.82	1.43	1.68
43	Low	4.25	5.26	5.88	2.86	3.26
44	High	5.31	5.18	6.92	7.23	8.57
45	High	4.48	4.11	6.32	8.14	7.15
46	Low	1.56	2.35	2.86	2.28	5.29
47	Low	1.37	4.66	5.22	1.39	5.13
48	Low	3.94	4.23	5.23	4.81	5.95

Appendix CC

Logo Study - Saw Blade Actions Types

Table 111 - Number of various types of actions used by each participant in the completion of Saw Blade.

Participant	Condition	Saw Blade Time	Saw Blade Success	Add	Modify	Move	Delete	Move + Delete	Move + Modify + Delete
1	Low	98	1	7	6	0	1	1	7
2	High	733	1	7	17	1	1	2	19
3	High	273	1	7	13	0	1	1	14
4	High	273	1	7	0	4	0	4	4
5	Low	159	1	6	1	2	0	2	3
6	Low	243	1	6	7	0	0	0	7
7	High	246	1	8	2	5	0	5	7
8	Low							0	0
9	Low	232	0	5	29	1	0	1	30
10	Low	121	1	5	4	1	0	1	5
11	High	216	1	7	15	0	0	0	15
12	Low	770	1	5	51	4	0	4	55
13	Low	29	0	1	2	0	0	0	2
14	High	283	1	6	4	0	0	0	4
15	High	83	1	6	3	2	0	2	5

Participant	Condition	Saw Blade Time	Saw Blade Success	Add	Modify	Move	Delete	Move + Delete	Move + Modify + Delete
16	High	137	0	6	3	0	1	1	4
17	Low	130	1	6	4	0	0	0	4
18	Low	163	1	6	5	0	0	0	5
19	High	249	1	14	16	0	1	1	17
20	Low	105	1	6	5	0	1	1	6
21	High	207	1	6	7	0	0	0	7
22	High	160	1	6	9	0	0	0	9
23	High	309	1	9	13	0	0	0	13
24	Low	924	1	29	28	6	10	16	44
25	Low	402	1	8	27	10	1	11	38
26	Low	444	0	9	28	0	4	4	32
27	High	96	0	4	4	3	0	3	7
28	High	460	0	10	73	3	2	5	78
29	Low	219	1	5	11	0	0	0	11
30	High	325	1	9	14	16	0	16	30
31	High	219	0	5	16	4	0	4	20
32	Low	58	1	5	2	0	0	0	2
33	Low	122	1	5	3	0	0	0	3
34	High	302	1	10	19	9	1	10	29
35	High		0					0	0
36	High	328	1	5	19	0	0	0	19
37	High	280	1	11	12	8	2	10	22
38	Low	294	1	9	7	13	4	17	24
39	Low	356	0	6	41	0	0	0	41
40	High	189	0	12	3	0	0	0	3
41	High	625	1	11	43	3	3	6	49

Participant	Condition	Saw Blade Time	Saw Blade Success	Add	Modify	Move	Delete	Move + Delete	Move + Modify + Delete
42	Low	166	1	5	7	0	0	0	7
43	Low	149	1	5	4	0	0	0	4
44	High	100	1	5	3	0	0	0	3
45	High	1131	1	6	58	0	1	1	59
46	Low	110	0	4	5	0	0	0	5
47	Low	393	1	7	6	0	0	0	6
48	Low	185	1	7	10	0	1	1	11