Simulating Human Users in Dynamic Environments

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Michael J. Schoelles
B.S., University of Maryland, June 1971
M.S., George Mason University, January 1994

Director: Henry Hamburger, Ph.D.
Professor, Department of Computer Science

Summer Semester, 2002
George Mason University
Fairfax, Virginia

Acknowledgements

I want to thank my committee for their tireless support and guidance. I am grateful for the complementary perspective on Cognitive Science that they have given me. Henry Hamburger and George Tecuci introduced me to AI and have provided me with a computational perspective. Wayne Gray and Len Adelman introduced me to Cognitive Psychology and have provided me with a human perspective.

I want to thank Wayne Gray and Debbie Boehm-Davis for bringing me into the Applied Cognition and Human Factors Laboratory (ARCHLAB). I am grateful to all the past and present members of the lab who have made this dissertation possible, especially Erik Altmann and Debbie Kranz.

Last and most of all, my wife, Kathy Arbogast, deserves special acknowledgment for her suggestions and support on the dissertation. She encouraged me to leave industry and pursue an academic career. She not only has made this endeavor possible, but a joy as well.

This work was supported by Air Force Office of Scientific Research Grant # F49620-97-1-0353.

Table of Contents

Page

List of Tables

Page

List of Figures

Page

Abstract

SIMULATING HUMAN USERS IN DYNAMIC ENIVIRONMENTS

Michael J. Schoelles, (Ph.D.)

George Mason University, 2002

Dissertation Director: Dr. Henry Hamburger

In the discipline of Human-Computer Interaction, user interface design engineers have tools such as interface tool kits that assist them in development of the computer side of the interaction, but are lacking in tools for the human side. A kind of tool that would be valuable to design engineers is one that would evaluate and test different designs by interacting with them as a human user would, exhibiting the same constraints and variability as human users. As a step towards fulfilling this need on the human side, I developed a Simulated Human User and a Simulated Task Environment, named Argus, for the judgment and decision making domain.

The contribution presented here extends modeling human-computer interaction beyond the typical laboratory domain to a complex and dynamic domain. The domain is complex in the sense that many aspects of the task are unconstrained, thereby allowing users to adopt different interactive strategies (i.e. different combinations of user actions) to accomplish a task goal. The domain is dynamic in the sense that the environment changes over time and there is a time pressure to respond to the changes.

The Simulated Human User is a computational cognitive model implemented under the ACT-R 5.0 embodied cognitive architecture. The model is the most comprehensive and successful ACT-R 5.0 model of human-computer interaction. The model is the first ACT-R 5.0 model to stress the importance of parallelism among cognitive, perceptual, and motor operations in HCI. This dissertation extends the ACT-R 5.0 architectural mechanisms to determine strategies for visual search at the cognitive level. It also implements changes to the visual attention mechanism of the vision module.

Argus is a family of Simulated Task Environments for performing decision making tasks. The design features extensive task environment configuration and control facilities. The system collects and integrates data at a finer grain size than other existing simulated task environments.

The model correctly predicts the overall performance decline on a decision task when workload is increased by adding another task. The results indicate that veridical simulation of human users in complex dynamic environments can be achieved at one level by capturing the range of strategies they employ at the next lower level. Incorporating a Simulated Human User into a design tool will enable user interface designers to make decisions based on quantitative results.

1. Introduction

Imagine that you are a Graphical User Interface (GUI) design engineer working on an interface for a complex control system like a nuclear power plant. You have a requirement to add a secondary task to the system. Several questions confront you. Will performance on the primary task degrade? If so, by how much? Are certain subtasks affected more than others? Can interface changes be made to the primary task that will lessen the impact of the secondary task? The process of making ad hoc design changes only to find out they do not work is very costly. The foregoing questions could be answered by a Simulated Human User (SHU) operating on a simulated environment. A simulated user is needed because the change must be tested under many different conditions with operators of varying skill levels and other cognitive differences.

The Human Computer Interaction (HCI) community is envisioning the use of such simulated human users in the form of computational cognitive models as design, testing, and usability tools for User Interface Engineers (Freed and Remington 2000; Myers, Hudson et al. 2000; Ritter, Baxter et al. 2000; Hornof 2001; Ritter and Young 2001). As a design tool, a cognitive model functioning as a SHU could provide qualitative and quantitative results as input to the user interface design decision making process. In the testing arena, replacing actual users with SHUs could speed up the testing cycle and reduce cost. SHUs can further reduce costs since they can be employed early in the iterative design process, and the earlier design problems are detected the less costly they are to fix. All of these potential uses involve understanding interactive behavior.

Figure 1 indicates that interactive behavior emerges from the relationship among an Artifact (i.e. the computer interface), the Task to be performed on it, and Embodied Cognition (i.e. the human). The embodied cognitive system consists of the cognitive (including memory), perceptual, and motor capabilities of the user.

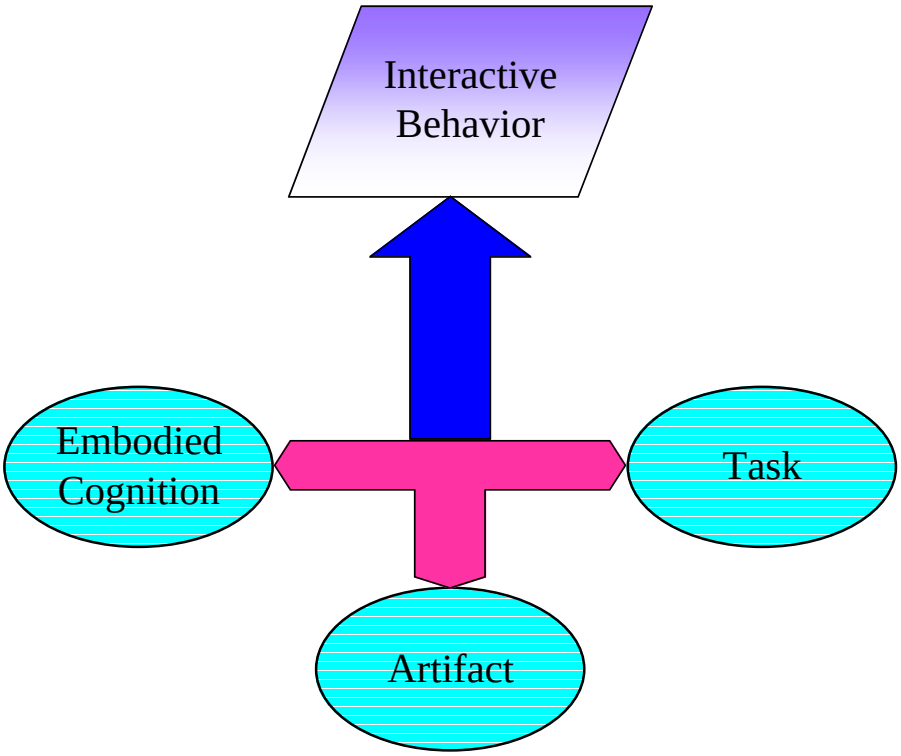


Figure 1. The Embodied Cognition, Task, Artifact (ETA) Triad. The interactive behavior of a human using an artifact to perform a task is a function of the constraints and opportunities emerging from the interaction of embodied cognition (cognition, perception and motor processing), the task and the artifact.

The central idea of the ETA framework (Gray 2000; Gray and Boehm-Davis 2000) is that the User Interface Designer must consider all three components individually or pairwise, as computer science and cognitive psychology have typically done in the past (Byrne in press).

Computer scientists have concentrated on the artifact, while cognitive psychologists have concentrated on cognitive constraints. Instantiating the embodied cognitive system as a computational cognitive model rather than as a human is the means to achieve this integration. The design of the artifact and the task can serve to constrain or permit different combinations of human behaviors. A simulated human user programmed with task instructions and acting in effect on the same computer interface (i.e. a representation at the pixel or object level of the computer interface) as the human allows exploration of these constraints and opportunities.

Applying Artificial Intelligence (AI) techniques to HCI to build SHUs is now possible due to a number of advances in Cognitive Science. To simulate a human doing a complex task requires a theory of cognition that considers all aspects of cognition (Newell 1990). Unified cognitive architectures such as ACT-R, SOAR, and EPIC fill this need and have been implemented as computer programs. To interact with a computer application requires that the SHU not only simulate cognition, but perceive and act as well. Theories of embodied cognition (Ballard, Hayhoe et al. 1997) posit constraints and opportunities among cognition, perception and action that can be implemented in a simulation. In conjunction with theories about perception and action, human simulation requires accurate timing data and location data about where humans look and point. Advances in measuring devices such as eye tracking equipment has improved collection and analysis of human data on which simulations can be based.

The central goal of cognitive modeling applied to the domain of HCI is to develop an easy to use interface design and testing tool that has as its core a cognitive model of the user that veridically simulates the user interaction with the system at a time scale of 300 ms. This is the time scale at which many cognitive, perceptual, and motor actions occur. For example, the time to move visual attention takes about 185 ms. The cognitive model when initialized with knowledge of the task and interface and subject(s) could accurately predict the consequences of interface

design decisions. One can imagine at a high level how this might be accomplished. The model might create the task knowledge by reading task instructions. Interface knowledge might be created by a representation of the graphical user interface obtained from an Interface Design Tool. The model would contain procedural knowledge about how to operate the interface object. The model would be capable of generating the strategies that humans use to do the task. The current state of the art is far from this ultimate goal. Currently, embodied cognitive models have been constructed that manipulate interfaces and perform interactive tasks using a limited set of strategies (Byrne 2001). The time scale of the interactions that can be analyzed is about 3 to 10 seconds. To close the gap between where we are and the goal requires progress on a number of technological issues including the following.

The underlying cognitive architectures are underspecified since they are works in progress. For example, a theory of visual scene analysis is a desirable capability of the architecture but has not been implemented in any of the main cognitive architectures.

The tasks that have been modeled for the most part have been relatively simple static tasks that isolate the phenomenon under study. Models must be developed to perform complex dynamic tasks. The model should be able to do all aspects of the task.

The models developed should be robust to changes in the interface or task. If the interface changes or another task is added, the model should be able to predict performance changes in the existing task.

Humans exhibit a wide range of behaviors in performing interactive tasks. An interactive task can be accomplished in a variety of ways. A particular way of doing a task is called a task strategy. The interface design may constrain the number of strategies or provide opportunities for unforeseen strategies. Strategy differences are a significant factor that must be addressed to account for human variability. Because strategies exist at different levels, an inherent feature of

human-computer interactive behavior is that it is compositional. A task is composed of smaller subtasks. These subtasks can be decomposed into yet smaller subtasks. At each level various combinations of subtasks are possible to achieve the task at the next highest level. Since different people bring different experiences and knowledge to the task at hand, they execute different combinations of the subtasks to achieve the higher level task strategies. Humans also vary in their cognitive and psychomotor abilities. Models of human behavior have not done very well in matching the variation demonstrated by humans.

This dissertation addresses some of the technological problems stated above. It is intended as a proof of concept that a computational cognitive model can make correct predictions about human computer interactive behavior in a complex dynamic environment. The model addresses the problem of accounting for human variability. The cognitive model presented here is the most complex ACT-R model that has been developed. The task environment, named the Argus STE, in which the model performs is the most complex and dynamic task environment of any ACT-R model. The generic model will be referred to as the Argus model. When this model is instantiated as a user it will be referred to as a SHU.

The Argus STE supports two types of task. The first is called the Classification task. This task is complex because it involves a combination of visual search and complex time critical decision making. It is dynamic because the environment continually changes. The second task is a highly perceptual and motor intensive task called the Tracking task. A dual task environment is created when this task is added to the Classification task. The Argus model is sensitive to changes in the interface and increased workload. Workload is increased by the addition of the Tracking task. The Argus model performs multitasking to process this increased workload.

The Argus model provides a level of analysis below the unit task level. Successful prediction of interactive behavior depends on solving the composition problem. The predictive

capacity of a model must be at a level that is relevant to the design process. For example, a model might make correct predictions about overall performance but this result may have been achieved by over-predicting a variable on one subtask and under-predicting a variable on another subtask.

The work I am presenting advances the predictive power of simulated human users in complex dynamic interactive environments. Specifically, the SHU will predict multiple-task performance imposed by different design features. The issue of multiple-task performance is closely tied to the issue of predicting and measuring the cognitive workload imposed by these design features (Wickens 1992). Previous modeling applied to HCI (Hornof and Kieras 1997; Kieras and Meyer 1997; Chong 1998; Byrne 2001) has focused on single task models that predict overall performance, and not at modeling the smaller steps that combine to produce overall performance. I propose to design and implement a SHU that models multi-task behavior at a smaller grain size in order to assess the impact of interface design changes on cognitive workload. The SHU will interact with a simulated task environment designed to study the effects of interface changes on interactive behavior under varying degrees of cognitive workload. The term simulated task environment refers to a system that retains enough of the complexity and characteristics of the system being simulated to achieve the research goal and abstracts away those features not essential to the research goal (Gray 2002). For my proposal, the simulated task environment is that of a radar operator. Part of this effort is to define an objective measure of cognitive workload, but as a first approximation, it is a measure of the total cognitive, perceptual, and motor effort involved in completing a task.

1.1. Overview of Dissertation

This dissertation first introduces cognitive architectures and describes three cognitive architectures that have been applied in the discipline of human–computer interaction. Special

emphasis is paid to the ACT-R architecture because it is the basis for the cognitive model developed in this dissertation. The next chapter presents related work in which researchers have applied cognitive modeling to human-computer interaction. The notion of a simulated task environment on which the Argus model operates is introduced. The Argus STE is described in detail along with the empirical studies that produced the human data. The structure and operation of the Argus model are specified. The actual code for the Argus model is given as an appendix. The results chapter presents the statistical analysis in order to validate the Argus model and demonstrates the predictive power of the model. The theoretical and applied implications for this work are discussed. The dissertation concludes with some ideas for future work.

2. Theoretical Background

2.1. Cognitive Architecture

The motivation for cognitive architectures in Cognitive Science was seeded by Newell when he proposed to the cognitive psychology community that you can't play twenty questions with nature and win (Newell 1973). He observed that explanations for phenomena such as subitizing, serial position effects, and continuous rotation effect were being framed as different binary oppositions such as nature vs. nurture, innate vs. learned, and parallel vs. serial. He also noted that these phenomena were being studied in isolation using simple tasks. He posed that this type of research will never result in understanding cognition, and suggested three courses of action. The first was to develop complete process models based on the production system framework. The second was to concentrate effort on a single complex task and to study all of its parts and how they fit together. The third was to develop a single system, a model of the human information processor, that could do all the diverse collection of small tasks.

In 1990, Newell put forth the challenge to develop a Unified Theory of Cognition (UTC), which he described as a single set of mechanisms to explain all of cognitive behavior (Newell 1990). A fundamental principle of AI is the physical symbol hypothesis, which states that a physical symbol system meets the necessary and sufficient conditions for general intelligence. Newell saw UTCs as a means to empirically and analytically prove the sufficiency requirement for general intelligence.

2.1.1. What is a Cognitive Architecture?

A cognitive architecture is a computer program that codifies or simulates the fixed mechanisms hypothesized by a Unified Theory of Cognition. A computational cognitive model, as shown in Figure 2, is developed in the modeling language specified by the architecture and executed by the cognitive architecture implementation. This language and program represent an implementation of a particular UTC. The architecture, since it is implemented as a computer program, provides an explicit and precise account of the theory. In 1990, Newell proposed a search for unification of cognitive theories, but in 2002 the goal still has not been achieved. The unified theories put forth are still evolving. As more cognitive models are built and validated in different domains, new data leads to changes in the theory and thus to different architectures. The Argus model is an ACT-R 5.0 architecture-based model.

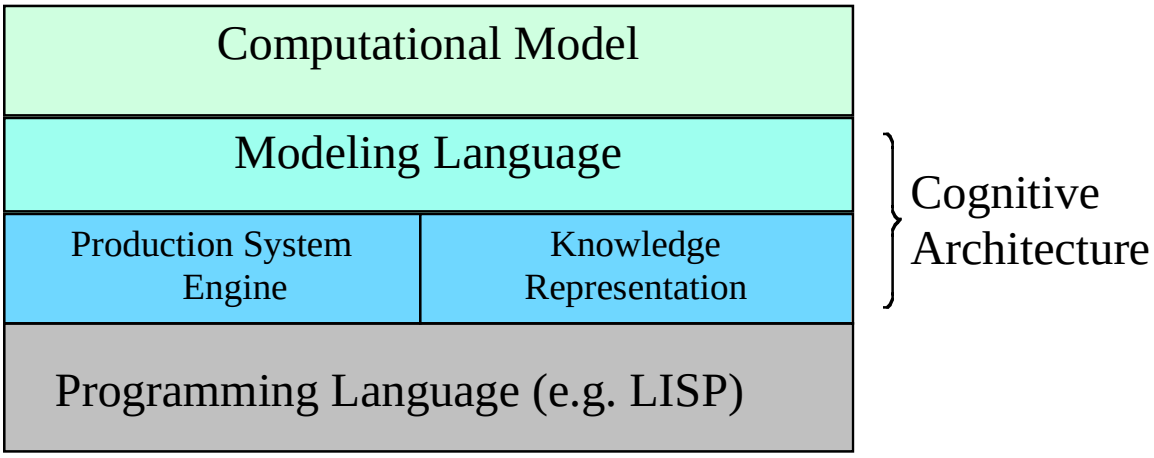


Figure 2. Computational modeling implementation framework.

Pylyshyn (1984; 1991) adds a constraint, termed cognitive impenetrability, on these mechanisms. He explains cognitive impenetrability as the invariance of certain low-level

cognitive functions to changes in higher level goals and knowledge. The functions are the primitives which produce behavior across domains and across cognitive faculties. They are the building blocks of cognition. Human-computer interactive behavior, by its very nature, requires more than just cognitive constraints. The perceptual and motor systems are also sources of constraints and have been incorporated into many cognitive architectures.

Details of the three most widely used cognitive architectures in HCI are discussed below. They share many of the same general assumptions about what mechanisms are required and they differ on the details of these general assumptions. All three architectures assume the physical symbol hypothesis.

Another unifying assumption of these architectures is that human behavior is goal oriented. Closely related to the assumption of goals is that a control mechanism is required. The production system framework accounts well as a control mechanism for goal oriented behavior.

2.1.2. Three Cognitive Architectures for HCI

Three architectures that have been applied to the HCI domain are Executive-Process/Interactive Control (EPIC), SOAR, and ACT-R 5.0. These architectures share some assumptions and have several features in common. Each architecture assumes the physical symbol hypothesis and is implemented within the production system framework. Each one is goal oriented (i.e. the flow of control is geared toward achieving a goal), contains a memory system(s) and interacts with the environment to some degree. All three specify a cycle time for production execution of 50 ms., which has provided good fits to human latencies over a wide range of tasks and therefore has become the commonly accepted value for cycle time. This section will describe each of the architectures in terms of these features and in enough detail to differentiate them. Their commonalities and differences are summarized at the end of the section in Table 2.

2.1.2.1. Executive-Process/Interactive Control (EPIC)

The EPIC cognitive architecture, developed by Kieras and Meyer (Kieras and Meyer 1996; Kieras and Meyer 1997; Meyer and Kieras 1997) was motivated by the need to reduce the design and testing time for computer interfaces. EPIC was designed specifically to simulate human performance in human computer interactions and to model multiple task performance. The design explicitly integrates perceptual motor mechanisms with cognitive mechanisms to produce procedural skill. A simulated task environment, which simulates a computer screen and interaction devices, is also included in EPIC. In addition to a cognitive processor, EPIC specifies separate perceptual processors for each sensory modality, each with its own timing characteristics. EPIC also specifies separate motor process for manual, oculomotor, and vocal processors. Feedback pathways from the motor processes and tactile feedback help to coordinate multiple tasks.

Memory is partitioned into production rule memory and memory for declarative information. Working memory contains all the temporary information needed for processing on each cycle. Working memory is divided into a control store, a general store, and stores for the perceptual motor processes.

EPIC uses the parsimonious production system (PPS). The production rules are condition action pairs. Conditions can only test the contents of working memory. Actions can add and remove items from working memory or issue a command to the motor processor.

The cycle time is normally fixed at 50 ms but the system does have a mode where the cycle time is stochastic with a mean of 50 ms and can vary by 20%. At the start of each cycle, working memory is updated with output from the perceptual processors. All productions are matched against working memory. The unusual feature of the PPS is that all productions whose conditions match working memory will fire. Thus EPIC simulates a parallel cognitive system at

the symbolic level. The intent of this design was to avoid an executive process bottleneck yet still achieve orderly processing by relying on memory and perceptual-motor constraints. At the end of each cycle, working memory is updated and commands are sent to the motor processor.

Working memory contains a control store to provide executive control over production execution. The control store contains memory elements that represent the current goals and current steps within the current task. The control store is unlimited in capacity and information is not lost from the control store. Other task information is kept in the general working memory store. EPIC does not include any assumptions about memory decay, capacity or representation.

The perceptual processors in EPIC are the visual processor, auditory processor and tactile processor. Only the visual processor will be described here because it is the most relevant to the Argus STE tasks. The visual processor simulates the retina of an eye. The type of information available about a visual object depends on the retinal distance between the object and the center of the fovea. EPIC divides the retina into three circular areas. The fovea area has a radius of one degree of visual angle, the parafovea has a radius of ten degrees, and the periphery has a radius of sixty degrees. The visual acuity of the object is highest in the fovea and lowest in the periphery. The visual processor has a working memory that is chained to the visual environment. That is, visual working memory is updated as changes occur to the visual object in the environment. The visual processor also may respond with outputs at different times for a single stimulus. The outputs include simple detection, feature resolution, and object recognition.

The motor processors in EPIC are the manual processor and the oculomotor processor. The motor processors receive commands from the cognitive processor and produce simulated movements with the appropriate timing characteristics. Both motor processors have the same general structure and represent movement generation and movement in the same manner. Movements are represented in terms of a hierarchical movement feature structure. Movement

timing is a function of the feature structure as well as the mechanical properties of the movement. Movement is divided into two phases, a feature preparation stage and an execution phase. The timing for the preparation stage depends on how many of the features that the movement requires are already in the required state. 50 ms is required for each new feature that must be prepared. The execution phase begins with a 50 ms delay to initiate the movement. The time to complete the movement depends on the part to be moved and the type of movement. In some cases, preparation of a movement can overlap with the execution of the previous movement.

The manual motor processor contains several movement styles. These include the punch style for punching a key or button when the key or button is below the finger, the peck style for key presses that may require some horizontal movement, posing a hand, poking at an object, pointing to an object with a mouse, and plying an object such as a joystick. Each style has its own feature structure. The manual processor in ACT-R 5.0 is closely patterned after the manual processor in EPIC.

The oculomotor processor implements voluntary and involuntary modes of movement. Voluntary movements are saccades to an object. They require two features, extent and direction. Saccades are executed at 4 ms per degree of visual angle. Involuntary movements are saccades triggered by the rapid onset of an object and small smooth adjustments in response to moving attention. The oculomotor processor can also track a moving object using a combination of voluntary and involuntary movements.

2.1.2.2. SOAR

The SOAR cognitive architecture, developed by Laird, Rosenbloom and Newell (Newell, Rosenbloom et al. 1989), can be traced backed to the General Problem Solver (GPS) (Newell and Simon 1963). A central idea of the GPS was that human intelligence can be cast as a search through a problem space. This idea was cast in the Model Human Processor (Card, Moran et al.

1983) as the Problem Space Principle. The search is guided by goals and takes actions to achieve these goals. In deciding which action to take an intelligent system will use all available knowledge. SOAR was a starting point for Newell's vision of a cognitive architecture as a "unified theory of cognition" (Newell 1990).

All tasks are goal-oriented and represented by problem spaces. A problem space is a set of states including the initial state and goal state. The space also contains a set of operators. Operators are applied to states to yield other states until the goal state is obtained.

SOAR is a production system with two types of memory, a production memory and a working memory. The production memory contains the productions represented as associations between conditions and actions. Conditions and action are represented by feature value pairs. All long term knowledge, declarative and procedural, is represented as productions. The working memory contains the short-term processing context. The primary structure in working memory is the goal stack, which contains the current goals in a hierarchy. Each goal contains the current state of the associated problem space and the current operator in that space if one exists.

Execution in SOAR, like EPIC and ACT-R is based on a production cycle. In SOAR it is called a decision cycle. On each decision cycle, SOAR attempts to determine how to change the current context. The change to the context can be either a new goal, a new problem space, a new state or a new operator. This decision cycle has two phases. In the first phase, called elaboration, all productions that match working memory fire in parallel. These productions can change working memory or create preferences. Preferences are suggestions about which operator to apply. As a result of changing working memory new productions can fire. The elaboration phase continues until no productions match the current contents of working memory. Thus the elaboration phase ensures that all available knowledge is brought to bear on the decision. At the end of the elaboration phase, no action has taken place since preferences only suggest operators.

In the second phase, called the decision phase, preferences added to memory are evaluated by a fixed domain-independent decision procedure. If the change to the current context is the selection of an operator, the operator is applied to the current state at the start of the next elaboration phase.

At the end of a decision cycle, it is possible that no decision could be made. When this occurs a special structure called an impasse is created. When an impasse arises, SOAR creates a goal to resolve the impasse. The architecture creates the subgoal but domain knowledge in the form of productions must fill in the rest of the context. For example, a production may suggest switching problem spaces. The impasse could also be resolved by new information coming in from perception.

SOAR learns new knowledge by forming new associations (i.e. productions). Impasses are formed due to lack of knowledge and therefore present opportunities for SOAR to learn. When the impasse is resolved a new association is formed between the pre-impasse environment and the actions that resolved the input. This new association is added to production memory. When the situation arises again this production can be executed directly. This process of learning by a model is called chunking and the learned productions are called chunks. The chunking process is deductive, automatic and ubiquitous. Although this deductive process is the only architecturally defined learning mechanism, other learning mechanisms have been built on top of the system's deductive mechanism (Johnson, Krems et al. 1994).

2.1.2.3. ACT-R 5.0

The architecture selected for this research is ACT-R 5.0 which combines ACT-R's theory of cognition (Anderson and Lebiere 1998) with a Perceptual/Motor component (Byrne 1999) (Byrne and Anderson 1998). The Perceptual/Motor component is extended to include Eye Movements and Movements of Attention (EMMA) (Salvucci 2000). The basic tenet of the ACT-R theory of cognition is that cognition adapts to the statistical nature of the environment. The

architecture is a hybrid architecture with a symbolic and a Bayesian sub-symbolic component for both procedural and declarative memory. The declarative memory system is activation based. The control strategy is a highly constrained production system. The perceptual/motor component is based on modal theories of visual attention (Anderson, Matessa et al. 1997) and motor movement (Kieras and Meyer 1997). The architecture specifies timing information for all three processes as well as parallelism between them.

ACT-R 5.0 is the latest version of the ACT architecture. In the early 1990's, rational analysis (Anderson 1990) was added to the ACT theory resulting in ACT-R 2.0 (Anderson 1993). This architecture has proven successful in the field of Intelligent Tutoring Systems. The LISP and Algebra tutoring systems were built using ACT-R 2.0. ACT-R 2.0 provided few constraints on the computational power of its production system framework and therefore was not able to account for data at a level below the rational band. To account for data at smaller time scales, constraints on production syntax and conflict resolution were added to the ACT-R architecture resulting in ACT-R 4.0. Next, the architecture was extended to incorporate perceptual and motor processes, resulting in ACT-R/PM.

ACT-R 5.0 brings to the ACT-R architecture a better integration of cognition, perception, and motor processes. It also adds some neural plausibility to the architecture. Figure 3 depicts the ACT-R 5.0 architecture. The remainder of this section specifies those features of ACT-R 5.0 that make it a useful architecture for HCI and that are essential to understanding the Argus model.

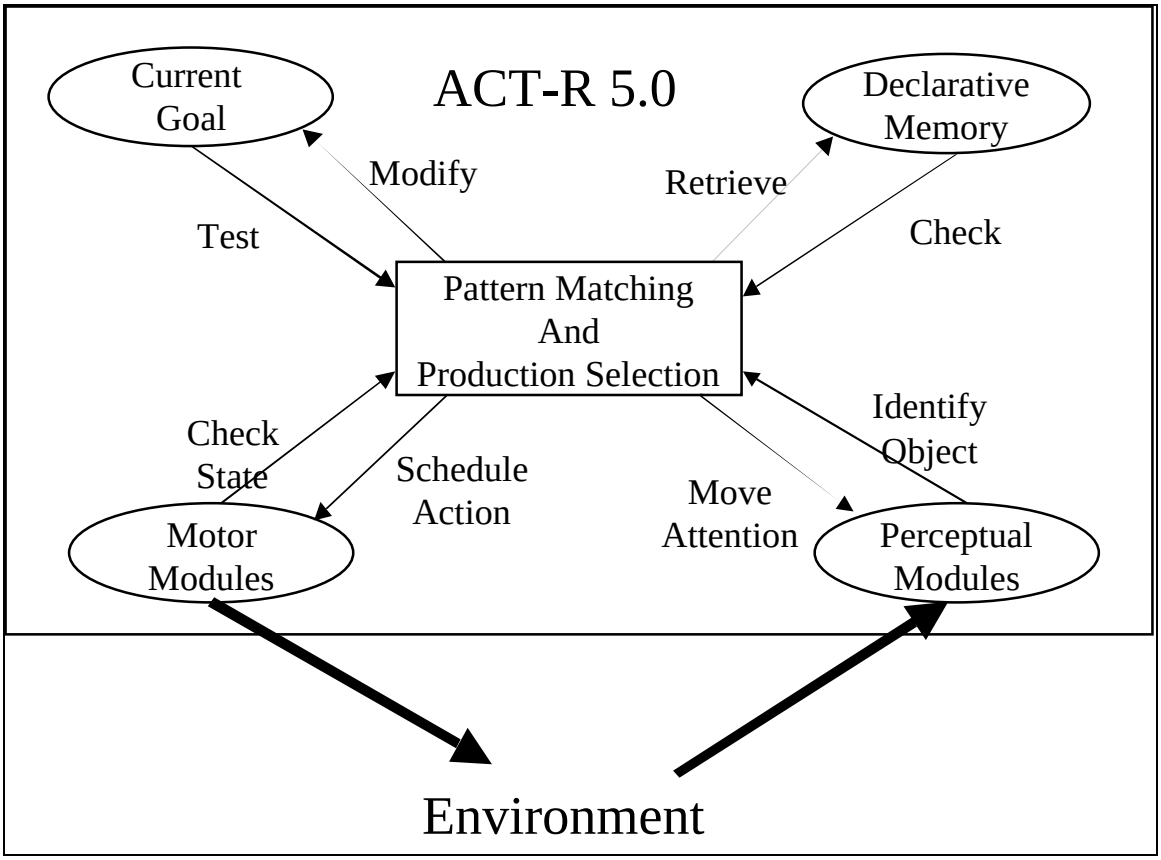


Figure 3. The ACT-R 5.0 Architecture (Anderson 2001).

2.1.2.3.1. Buffer Architecture

The buffer architecture design is highly beneficial in developing models of HCI. It enables the asynchronous operation of the memory system, the visual system, the auditory system and the motor system. In ACT-R 5.0 a memory retrieval is initiated as a request to the memory system on the right hand side of the production in the same manner as perceptual and motor actions are initiated. ACT-R 5.0 contains a set of buffers for declarative memory, vision, audition, motor processes and speech. Constraint satisfaction is carried out through these buffers. That is,

the conditions specified on the left hand side of a production rule are matched to the current contents of the specified buffer. The production shown below illustrates this point.

```
(defp hook-target
=goal>
        isa     select-target
        step    moved-cursor
=manual-state>
        isa     module-state
        modality free
=retrieval>
        isa     target
        tn      =tn
=visual>
        isa     screen-targ
        tn      =tn
!eval! (cursor-over-target-p =tn)
==>
+manual>
        isa     click-mouse
=goal>
        step    after-hook)
```

The left hand side (LHS) of this production (i.e. the lines above the ==>) are interpreted as follows. If the goal buffer contains a memory element of type select-target and its step slot is moved-cursor and the manual-state buffer contains a memory element of type module-state and indicates that the motor system is free and the retrieval buffer contains a memory element of type target with some value in the track number (tn) slot and the visual buffer contains a memory element of type screen-targ and its track number slot matches that of the retrieval buffer and the cursor is over the target then this production can be placed in the conflict set. The LHS of all productions in procedural memory are matched to the current state of all the buffers. All that match are placed in the conflict set where the one with the highest utility will fire.

The right hand side (RHS) of a production initiates changes to buffers. A buffer change can be either a direct setting of a slot in the memory element contained in the buffer or a request to launch a command to the process associated with the buffer. A request to a process is indicated by a + in front of the buffer name. In the example above, a request to the motor process to execute a mouse click is shown and a direct change to a slot in the memory element occupying the goal

buffer is also shown. The actions that are relevant to the Argus model presented in this dissertation are shown in Table 1.

Table 1. ACT-R 5.0 buffers and actions that occurs when a buffer change is issued on the right hand side of a production.

Buffer	**Module**	**Action**
goal	cognition	Replace contents with new memory element (create new element if specified)
retrieval	cognition	Retrieve from declarative memory the memory element with the highest activation that matches the pattern specified
visual-location	vision	Find a location on the screen that matches the pattern of features specified. Create a location memory element and put in buffer
visual	vision	Move visual attention to the location specified. Create a visual object memory element from the features found at that location. Put the memory element in the buffer
manual	motor	Execute the action such as move-cursor, a mouse click or key press

2.1.2.3.2. Vision Module

The Vision Module of ACT-R 5.0 provides the capability to execute pre-attentive feature searches (discussed below), shift the visual focus of attention, create declarative memory elements from the environment, and track the movement of the object that is the current focus of visual attention. As indicated above, the Vision Module manipulates the visual location buffer, visual attention buffer, and the visual state buffer. A model requests an action on the right hand

side of a production. The module executes the action and places the result in the appropriate buffer. A model checks for completion of the action on the left hand side of some production.

The Vision Module is based on the location-based movement of attention framework. The operation of the vision module is based on the hypothesis that certain features of an object can be located without attending to the object. To encode the object itself requires visual attention.

2.1.2.3.2.1. Feature search

The Vision Module feature search mechanism is based on feature integration theory (Triesman and Gelade 1980).

Features are salient properties that can be located in the visual scene without focusing visual attention at that location. A feature search request to the vision module instructs the module to find a location in the visual scene that meets a single constraint or a conjunction of constraints. The constraints include location, color, size, kind, plus a modeler defined value. The module searches its representation of the visual scene for the constraints specified, and if successful, creates a declarative memory element of type visual-location which contains the location in the visual scene that satisfies the request. It places this element into the visual location buffer. If unsuccessful it places the failure declarative memory element in the visual location buffer. A model can test the visual-location buffer for successful completion of the request and use the visual location memory element contained in the buffer for other perceptual or motor functions.

2.1.2.3.2.2. Movement of Visual Attention

One use of the visual location memory element is as an argument to the move visual attention command. The Vision Module moves attention to a location and creates a declarative memory element representing the object at that location. In pure ACT-R 5.0 the time required to

move and encode the object is a parameter with the default set to 85 ms. The Argus model uses Eye Movements and Movement of Attention (EMMA) (Salvucci 2000) in addition to ACT-R 5.0. EMMA integrates eye movements into the visual attention process. Encoding time under EMMA is based on how often the object has been encountered and encoded and distance of the object from the fovea. The visual memory object created contains the location and properties of the object and is placed in the visual buffer so that a model can check for completion of the action.

2.1.2.3.2.3. Visual Module State

The state buffer for the Vision Module can be checked by a model to determine if the module is free or busy.

2.1.2.3.3. Motor Module

The Motor Module in ACT-R 5.0 is based on the Motor Module of EPIC (Kieras and Meyer 1996). The Motor Module receives commands through the manual buffer. The commands that a model can issue to the Motor Module are to move the mouse, click the mouse, or press a key on the keyboard.

A motor operation has two phases, a preparation phase and an execution phase. The idea of the preparation phase is to account for the time it takes it prepare the muscles for movement.

2.1.2.3.4. Memory retrievals

Declarative memory retrieval is based on the sub-symbolic activation function:

$A_i = B_i + \Sigma\omega S_{ij} + \text{noise}$ where B_i = base level activation, $\omega_j = 1/j$ * total source activation, S_{ij} = associative strength, and noise is from a double exponential distribution, i = target chunk, j = slot in goal

The base level component is a function of how often and how recently a memory element has been used. The more recent and the more often an element is retrieved the higher the

activation. Thus it is possible to purposely remember something for future use by rehearsal (i.e. retrieving an element over and over again some number of times). The ωS_{ij} is a strength of association between an element in the goal and the memory element being retrieved weighted by a portion of the total source activation available. Total source activation is the amount of activation available to be spread from the goal to the target element. In ACT-R 5.0 the total source activation defaults to zero.

In ACT-R 5.0, each retrieval requests is initiated on the RHS and completed at some later time depending on its latency. During this time, other productions may fire that initiate perceptual and motor operations or the production requesting the retrieval may initiate perceptual or motor actions. Thus, the cognitive, perceptual and motor processes may be interleaved.

2.1.2.3.5. Conflict Set

In ACT-R 5.0, for a production to enter the conflict set, for each buffer pattern specified on the LHS the corresponding buffer must match. Productions can enter the conflict set independent of the goal, since productions are not required to specify a goal pattern.

The conflict set may contain one or more productions. The selection of which production to fire is based on the Expected Gain equation.

$E = P * G - C + noise$ where E = expected gain, P = probability of success, G = value of the goal and C = cost of production

The probability of success is based on the ratio of the number of past successes to the total number of times the production has fired. The value of the goal and cost is expressed in units of seconds. The value of the goal is a parameter that for virtually all ACT-R models has been set to 20. The cost of the production reflects the effort to achieve the goal. The noise adds stochasticity to the conflict resolution process.

2.1.2.4. Summary

Table 2 compares ACT-R, EPIC, and SOAR. ACT-R 5.0 was chosen for the basis of the Argus model for the following reasons.

Table 2. Comparison of ACT-R, EPIC, and SOAR on features relevant to HCI.

	ACT-R	**EPIC**	**SOAR**
Type	Symbolic + sub-symbolic	Symbolic	Symbolic
Memory	Declarative Procedural	Working Memory Declarative Production	Production
Goal Representation	Goal Buffer	Control Store	Goal Stack
Perceptual Motor	Vision Module Motor Module (manual) Emma (eye) Auditory Module	Perceptual Manual Oculomotor	Link
Interaction with Environment	Screen Features Mouse Commands	Simulated	None
Source of Variability	Sub-symbolic noise	Production Cycle Time (optional mode)	None
Support	Extensive tutorials, documentation, mail list, published models, source code and development environment	Operation manual and source code	Tutorials, documentation, mail list

The executive control system (i.e. production execution), the memory system, the visual system and the motor system operate asynchronously which permits interleaving of human cognition, perception and action. Second, it can directly interact with the Argus task environment in the same manner as humans which is not possible in SOAR or EPIC. ACT-R 5.0 has a sub-symbolic layer which provides parallelism and variation in cognitive processing, while EPIC and SOAR are purely symbolic. ACT-R has a more extensive support system for model and theory development than either SOAR or EPIC.

2.2 Computational Cognitive Modeling and HCI

This section discusses past and present work that applies cognitive modeling to HCI. In particular, models that share some characteristics of the Argus model developed in this dissertation are discussed.

2.2.1. Foundations

The Goals Objects Methods and Selection rules (GOMS) (Card, Moran et al. 1983) model introduced a formal way to consider human constraints in HCI. For routine tasks, GOMS has proven very successful (Gray, John et al. 1993).

The notion of being able to have a computer program simulate a user was first put forth by Young (Young, Green et al. 1989) as a Programmable User Model (PUM). A PUM contained psychological constraints. A user interface engineer could program a user to perform a range of tasks on a target interface. One of the motivations behind this approach was to involve the designer in predicting usability and to make the designer cognizant of human constraints. Young (Young and Whittington 1990) used a PUM to predict conceptual errors in device usage with the intent that failures should be interpreted by the designer as pointers to deficiencies in the interface design.

GOMS and PUMS provided the foundation for cognitive modeling in HCI. The next step was the development of computational cognitive architectures. These architectures coupled with the new emphasis on embodied cognition has opened up new possibilities for the application of cognitive modeling to HCI (Ritter and Young 2001). Computational cognitive models that make predictions about human performance on interactive tasks have been developed in EPIC, ACT-R, ACT-R/PM and SOAR.

2.2.2. Predictive Engineering Models

A predictive engineering model is a model that predicts one or more behavioral measures independent of empirical data. To provide useful models of users for user interface designers, they must be predictive.

One of the first predictive engineering models of an interactive task was developed in EPIC (Kieras, Wood et al. 1995) (Kieras, Wood et al. 1997). This effort investigated the extent to which useful and accurate predictions could be made by EPIC models that were based on a priori task analysis and principles of construction. The task was a telephone operator task that typically spanned 0 to 16 seconds. The interaction was multimodal in that it required speech and keystrokes. Several models were constructed to provide different accounts of how the interactions were interleaved to produce rapid performance. The model was able to accurately predict total task time.

Kieras (Kieras, Wood et al. 1995) defines the goal of predictive engineering models as prediction of performance independent of empirical observations. To achieve this goal, he asserts, requires an accurate cognitive architecture and a means to choose and represent veridical task strategies. He developed the notion of modeling policies as a means to choose and represent task strategies.

Kieras (Kieras, Wood et al. 1995) also states some modeling issues for constructing a priori models, particularly issues concerned with motor movement. These issues are coordination within motor modalities, coordination between motor modalities, and movement anticipation. Modeling policies are different ways to resolve an issue. For example, within a motor modality, one policy could be to complete all movement phases before executing the next step vs. preparing the next movement while the current movement is executing.

2.2.3. Visual Search

A major component of the Argus model is visual search. The research presented below focuses on visual search in HCI for simple interactive tasks.

Hornof has developed a series of EPIC models of visual search in pull-down menus (Hornof and Kieras 1997). The first model investigated random vs. systematic search strategies in randomly ordered menus, concluding that both strategies are used. The second and third models investigated the interaction of cognition, perception and manual processes in the search of ordered pull-down menus. The models showed that people anticipate the menu item position and make their initial hand and eye movement before the item appears. Gray (Gray, Schoelles et al. 2000) reached the same conclusion in an ACT-R/PM model of a continuous dynamic task.

Hornof (2001) sees predictive engineering cognitive models as the basis of a user interface design tool for visual search. The tool would predict execution times based on a definition of screen layout and the visual search task. The practical goal for this type of tool is automatically predicting usability of screen layouts and web pages.

Byrne (2001) developed an ACT-R/PM model of the EPIC randomly ordered menu selection model discussed above. This work seeks to investigate the underlying perceptual/motor constraints as well as cognitive constraints involved in the task. The model starts with the search strategies implemented in the EPIC model. Eyetracking data is used to formulate additional

search strategies. The performance of the model improves with the addition of these new strategies. If a model incorporates many strategies, when to invoke a particular strategy is partially a function of the environment. The problem for a cognitive model becomes how to learn this relationship between a strategy and the environment. Byrne provides a methodology to solve this problem. He runs the model in a learning mode. In this mode the model is able to learn the costs of each strategy. These learned costs are then used to initialize the model when doing the entire task. The model also demonstrated that strategies which minimize local costs are preferred over those that minimize global costs.

2.2.4. Multiple and Dynamic Tasks

Another characteristic of the Argus model which is currently being investigated by other researchers is modeling HCI in dynamic environments and in dual task environments.

Salvucci (2001) is developing a model of an automobile driver. The driver model executes lane keeping, curve negotiation and lane changing tasks. The model utilizes a two level perceptual control mechanism. Cognitive processes are responsible for situational awareness and maneuver initiation and termination. The model successfully predicts attentional demands.

Multitasking is an issue when a secondary task of operating an in-vehicle device is added to the primary task of driving. Salvucci has developed a model that predicts driver distraction when a cellular-telephone is added. The model integrates the driver model with a separate model of cellular-telephone use. Two studies (Salvucci 2001; Salvucci and Macuga 2001) have been completed that validate this approach.

The Kanfer-Ackerman Air Traffic Controller (KA-ATC) task environment, like the Argus STE is dynamic (Lee and Byrne 1999). A user acts as an air traffic controller managing the landing of incoming planes. The user accepts planes from a queue and places them in a hold pattern, moves the planes within three hold levels and lands the planes on one of four runways.

Interaction is through cursor keys and a text based graphical display. The dynamic aspect of the task is that the planes can run out of fuel and the wind direction can change affecting the choice of runway. Lee and Anderson (2000) developed an ACT-R/PM model of skilled performance in the KA-ATC task.

Their methodology was to decompose the KA-ATC into a hierarchy of unit tasks, functional levels and keystrokes. Their goal was to fit the model to human performance based on task completion time and number of attention shifts. Their model is able to do the task using a single strategy. Lee and Anderson attribute differences in unit task times between the model and humans to the fact that the humans were using a greater number of strategies.

Their effort also pointed to a deficiency in the ACT-R/PM architecture. The architecture does not support bottom up processing of visual cues. That is, visual screen changes are not automatically detected by the vision module. The model must poll (i.e. check) at the cognitive level (i.e. a production) the vision module's iconic memory.

Another model of a dynamic environment in ACT-R 4.0 is currently underway (Gluck, Ball et al. 2002). This effort is attempting to model an Uninhabited Air Vehicle (UAV) operator. The model is interfacing to a UAV Synthetic Task Environment. The model is based on a unit task decomposition of the operator task. The unit tasks are to control the heading, altitude, and speed of the UAV.

2.3. Embodied Cognition and HCI

In order to successfully model human behavior under the ETA framework, the interaction of cognition, perception and motor processes must be incorporated into the SHU at a fine-grained level. What is fine-grained in this case? Ballard proposes 300 ms as the time scale at which perceptual and motor constraints interact with cognition, or the embodiment level (Ballard,

Hayhoe et al. 1997). At this time scale, information in the world can be made available to cognitive processes through perception, and cognition can act on the world through motor processes. Since information in the world can be accessed at this time scale, cognition is able to use the world as a memory device. This can serve to reduce the computational complexity of cognition because a complete real-time description of the world is not required. A partial description that is filled in as needed will do. This filling in of information will influence how a person interacts with the world at longer time scales. For humans interacting with computer displays, scan paths and search times emerge from functional visual behavior at the 300 ms level. For example, speeds of body movement indicate it takes 200 –300 ms to generate eye movements (Ballard, Hayhoe et al. 1997).

Gray & Boehm-Davis (2000) apply the basic concepts of embodied cognition to human-computer interaction. They argue that different interactive technologies support different ways of combining basic interactive activities or microstrategies, and that these microstrategies can be affected by small changes in interface design, and since these are routine activities over time there can be a significant cumulative effect on performance. The GOMS approach to behavior decomposition elucidated by Gray and Boehm-Davies is well-accepted as a task analysis tool, but the exponential explosion of the composition problem requires a computational solution.

2.4. Simulated Task Environments

> In field research, there is often too much complexity to allow for any more definite conclusions, and in laboratory research, there is usually too little complexity to allow for any interesting conclusions (Brehmer and Dörner 1993).

Figure 1 showed interactive behavior emerging from the interaction of the interface, embodied cognition, and the task to be performed. In HCI research it is not always practical to work with real world systems. This is often due to lack of availability or safety reasons such as

with nuclear power plant systems. In real world systems the source code is often not available or can not be changed for a variety of reasons. Often it is not even desirable to conduct research on the target system because of the complexity of the system. A complex system may be extremely large and require resources not available to the researcher or the system may require extensive training in order to operate.

In order to study interactive behavior in complex environments, one approach that has been successful is to work with some form of Simulated Task Environment (STE). A STE provides an environment that enables the researcher to control the complexity of the interface and task. Gray (Gray 2002) lists four types of simulated task environments: hi-fidelity simulators, scaled worlds, synthetic task environments, and simple laboratory environment. STEs vary in the control the researcher has over the simulation, how closely the simulation corresponds to the target system and how engaged users become. Hi-fidelity simulators attempt to simulate all aspects of the target as engagingly and realistically as possible. Scaled worlds focus on a subset of the functionality of the target system. Scaled worlds require extensive knowledge of the original task by the subject. Researchers intend their results to generalize back to the target environment. Synthetic task environments are focused more on theory and a particular kind of task. For example, a synthetic task environment may be constructed to simulate systems that require many rapid shifts of attention. Results are intended to generalize to classes of systems rather than to a specific application.

In HCI, a synthetic task environment aims to maintain the relationships between the interactive task, the interface design, and embodied cognition. The results of research done on these synthetic task environments can be formulated in terms of interface design guidelines for real world systems. Creating models of embodied cognition in synthetic task environments rather than laboratory environments allows researchers to investigate the relationships between interface

design, task, and the human condition. The Argus model and the Argus STE are constructed to investigate dynamic time critical interactive systems.

3. Argus Simulated Task Environment

The Argus STE facilitates research of human-computer interaction in the judgment and decision making domain (Schoelles and Gray 2001). The Argus STE was designed after an extensive investigation of similar simulated task environments. The systems investigated include Space Fortress (Donchin 1995), Advanced Cockpit (Ballas, Heitmeyer et al. 1992), the Team Interactive Decision Exercise for Teams Incorporating Distributed Expertise (TIDE2) (Hollenbeck, Ilgen et al. 1995; Hollenbeck, Sego et al. 1997), and Tandem (Dwyer, Hall et al. 1992). Like the Advanced Cockpit and Space Fortress, the Argus STE places a premium on embodied cognition (Kieras and Meyer 1996) and rapid shifts in serial attention (Altmann and Gray 2000). Like TIDE2 and Tandem, the Argus STE emphasizes judgment and decision-making in a multiple-cue probability task (Gilliland and Landis 1992). The Argus STE was designed to facilitate the investigation of a broad category of research questions centered on how interface design affects cognitive workload in tasks involving cognitive as well as perceptual-motor operations for both team and individual performance.

The Argus STE is a family of STEs. The single user version, named Argus Prime, is the version used in this dissertation. A team version, named Team Argus, has been used to study the effects of time pressure on team judgment and decision making performance (Adelman, Henderson et al. 2001; Adelman, Miller et al. in press).

Beyond the simulation, the Argus STE provides a suite of tools for creating task variations, manipulating experimental design, and collecting and analyzing data. The Argus STE

is highly interactive, requires rapid shifts in serial attention, incorporates a judgment and decision-making task and provides tools for the experimenter to systematically vary workload.

3.1. Hardware Requirements and Software Architecture

The Argus STE hardware and software architecture is shown in Figure 4. The software architecture is expanded in Table 3.

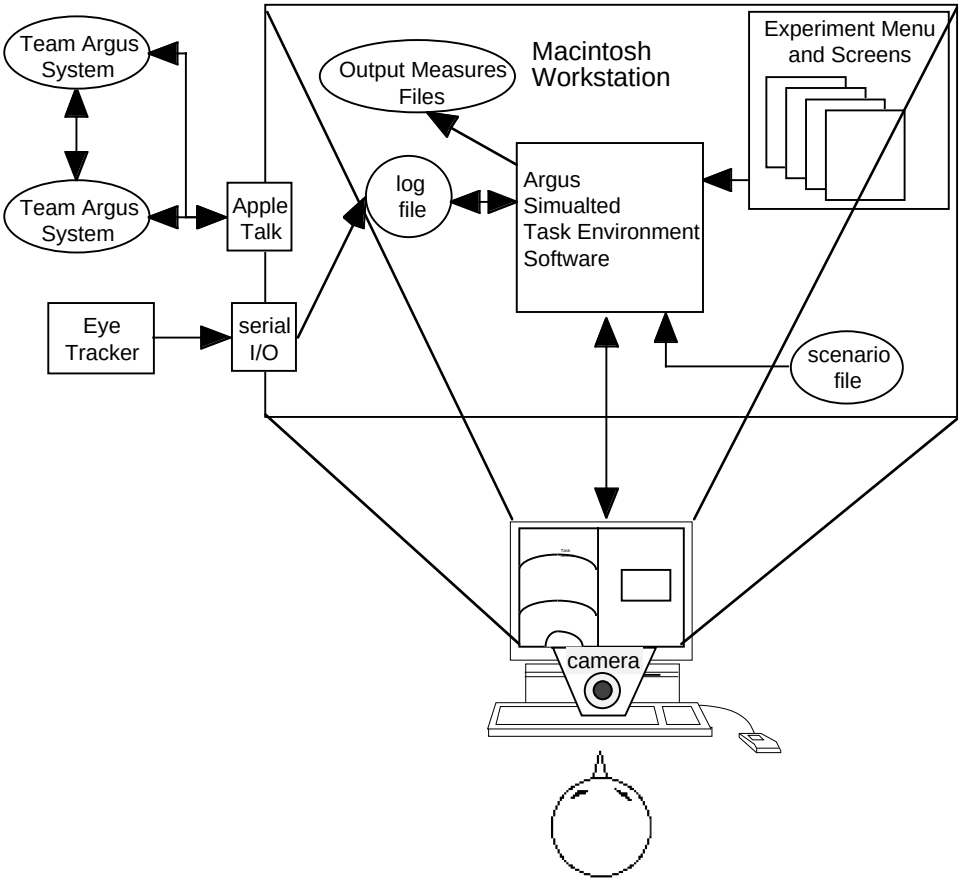


Figure 4. The Argus STE architecture. The eyetracker, Macintosh computer, camera, mouse and monitor are the hardware components. The Team Argus systems are connected via Appletalk. Files and interface layouts are shown in the ovals.

Table 3. The Argus STE software modules with functional descriptions.

Software Module	**Functional Description**
Event Module	Executes the events specified in the scenario at the proper time.
Task Interface Module	Displays targets on radar screen and updates information window.
Experiment Interface Module	Allows experimenter to configure and control system.
Data Analysis Module	Processes log file to aggregate and analyze data and writes results to output measures file.

For Argus Prime, the basic system runs on a Power Macintosh under the Macintosh Operating System and does not require special hardware or software. As shown in Figure 4, the configuration can support an Applied Science Laboratory E5000 Remote Eyetracker. The camera for this eyetracker sits under the display directly in front of the subject. Data from the eyetracker are input to the system through the Serial I/O port. For Team Argus, three Power Macintoshes are required, one for each team member. The team members are located in different rooms. Each computer is connected through the local AppleTalk network. A keyboard and mouse are used as input devices.

The Argus STE software system contains four modules. The Experimenter Interface is a GUI interface that allows the experimenter to configure the system to the particular experiment being run. The Event module executes the experimental session scenarios. The Task Interface module displays the scenario events to the subject and processes the subject interactions. Finally, the Data Analysis module creates a file containing all system and scenario events, as well as all subject actions (i.e., mouse clicks, mouse movements, and eye movements).

3.2. Classification Task

The subject's goal is to assess the threat value of each target in each sector of a radar screen. Over the course of a 12 to 15-min scenario, an individual using Argus Prime may classify 70-85 targets. As shown in Figure 5, the screen represents a portion of an airborne radar console with the subject's own aircraft, called "ownship" at the bottom. The arcs divide the screen into four sectors; each sector is fifty miles wide. The task is dynamic since the targets have a speed and course. A session is scenario driven; that is, the initial time of appearance, range, bearing, course, speed, and altitude of each target are read from an experimenter-generated file. The scenario can contain events that change a target's speed, course, or altitude. New targets can appear at any time during the scenario.

The subject selects (often termed hooks) a target by moving the cursor to its icon (which is a number called its "track number") and clicking. When a target has been selected, the information window (shown in Figure 5) appears in the right half of the screen (to the right of the radar shown in Figure 5). This window contains the track number of the target selected and the current state of the target. Target state is represented by the attributes displayed in the upper part of the window. Speed is the speed of the target in miles per hour (0 - 5000). Range is the distance in miles (0 - 200) from ownship. The bearing is the orientation of the target from ownship in degrees (0 - 360). Altitude is the altitude of the target in feet (0 - 75000). Course is the heading of the target in degrees (0 - 360). Approach distance is an estimate of how close in miles (0 - 300) the target will come to ownship. The subject is taught to convert the attribute values onto a 4-point scale with values 0,1,2, and 3 called cue values. Zero is the least threatening cue value and 3 the most threatening.

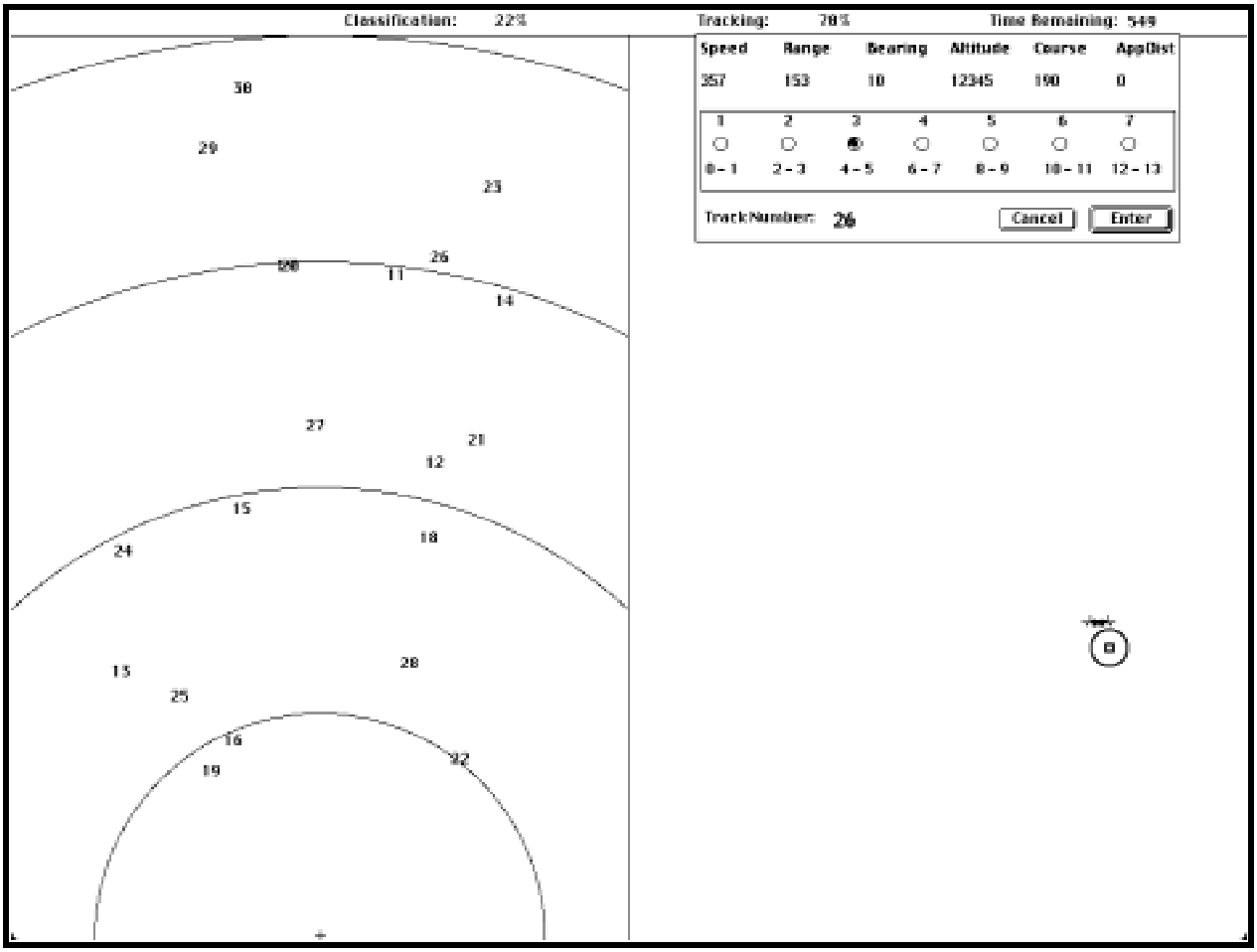


Figure 5. Argus Prime radar screen (left) and information window (upper right). The left half shows the radar screen and status bar (top). Ownship is indicated by the small '+' at the bottom. At the top is the summative feedback shown for classification and tracking tasks, as well as the time remaining in the scenario (in seconds). The top right half shows the target's attributes and the radio buttons used for entering the estimated threat value of the target on a 1-7 scale. The black dot in radio button #3 indicates the threat value assigned this target by the subject. The lower right half shows the tracking task; the subjects' task is to use the mouse to keep the circle with the square in it over the plane.

The score for a target is a weighted combination of cue values of a subset of the attributes. In a typical experiment the relevant attributes are range, altitude, and approach distance with weights 2, 1, and 2 respectively (e.g., a cue value of 3 for range would be multiplied by 2 to

yield a weighted value of 6). Help for each attribute is available by clicking on the attribute labels shown in Figure 5.

Figure 5 shows that the target score is mapped to a 7-point threat value by the subject selecting the radio button that corresponds to the threat value. This threat value is input to the computer by clicking on the Enter button. If the subject correctly classifies the target, full credit is given. If the subject incorrectly classifies the target, zero points are given. Targets must be classified once for each sector that they enter. If a target leaves a sector before the subject can classify it, it is considered incorrectly classified and a score of zero is assigned.

Feedback to the subject takes two forms. Summative feedback shown at the top of Figure 5 is given by a percentage. This percentage is updated each time a target crosses a sector boundary. It represents cumulative performance over all targets. Immediate feedback is an optional feature that can be controlled by the experimenter. If immediate feedback is enabled, the dialog box shown in Figure 6 appears when the subject finalizes the decision by pressing the Enter button. The dialog box contains a blue “thumbs up” icon if the decision was correct or a red “thumbs down” icon if the decision was incorrect.

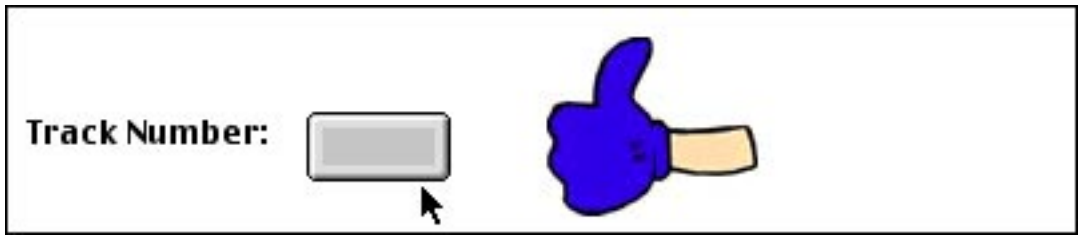


Figure 6. Argus Prime immediate feedback. A blue thumb up indicates the classification was correct. A red thumb down indicates an incorrect decision.

A log file is created for each scenario completed by a subject. The file contains all the data necessary for the analysis including each subject action, the information necessary to reconstruct what was on the display at the time of each interaction, eye-tracking data at 60 samples per second, and mouse position data for each eye track sample.

3.2.1. Interface Variations

A number of experimenter controlled features that vary workload as a function of the interface and task are part of the Argus STE design. A ubiquitous feature of the task is to determine if a target has been previously classified. Two interface variations have been implemented in Argus Prime to vary the effort required in this step. The first variation is whether or not to display the last classification decision made by the user, when the user selects the target. If the experimenter has set the condition to display the last classification made, the corresponding radio button is filled in. This variation is called the Dot interface after the dot in the radio button. The Dot interface permits determining the classification status of a target through a mouse action and a perceptual action. If the dot is not in use then the subject's determination of the target's classification status is purely a memory operation. A second variation is whether or not the immediate feedback for the classification task is accessible by the user. In the "feedback " (fb) condition, the feedback dialog box is masked with a gray window. To view the feedback the user must click on the gray window to uncover the feedbox underneath. Thus at least a mouse movement and an eye movement are necessary to process the information.

3.2.2. Multitasking

To increase perceptual and motor workload a multitasking environment is implemented in Argus Prime by adding a secondary tracking task. This task is based on the tracking task developed by Ballas (1992). The objective of the tracking task is to follow a continuously moving

target with a pointing device. The pointing device can be a joy stick or a mouse, but in the experiments conducted to date only the mouse has been used. The target is an icon of a plane. The mouse controls a tracking cursor which is a circular icon with a square in the middle. The subject switches from the classification task to the tracking task by pressing and holding down the shift key. This changes the cursor from its normal arrow shape to that of the tracking cursor. The subject then moves the mouse to keep the cursor over the plane. The plane moves in a random manner so the subject can not anticipate the movement of the plane. The subject switches back to the classification task by releasing the shift key. The shift key acts as a toggle between the classification task and the tracking task. The plane continues to move while the subject is doing the classification task.

A performance score is continuously calculated for the tracking task and displayed on the status bar next to the classification score. The performance score is a cumulative score based on incremental ratings that are calculated every tracking cycle. A tracking cycle is five seconds. The incremental ratings are shown in Table 4 and are based on the mean distance between the tracking cursor and the plane in pixels over the last five seconds.

Table 4. Tracking task incremental ratings indicated by the color of tracking cursor.

Tracking cursor color	**Tracking State (meaning of color)**	**Mean distance between tacking cursor and target plane in pixels**
Blue	Normal	< 50
Yellow	Warn	50 – 100
Red	Severe	> 100

The total performance score is the percentage of the total number of tracking cycles in which the cursor was blue. The score is calculated even when the subject is doing the classification task. The tracking task instructions direct the subject to try to keep the classification score and the tracking score equal.

4. Empirical Studies

The human data against which the Argus model is compared comes from the fourth (AP4) and fifth (AP5) Argus Prime experiments. The results of the first two Argus Prime experiments (AP1 and AP2) indicated that a recurring difficulty of the task was keeping track of which targets had been classified. In these experiments there was nothing on the radar screen to indicate whether a target had been classified; that is, when a classification was made, its on-screen icon did not change. However, in both studies, if an already classified target was reselected, that fact was indicated by one of the radio buttons in the information window. For example, if earlier in the same sector, the subject had assigned the target a threat value of 6, that radio button would still be highlighted (see Figure 5) when the target was reselected. When the target is no longer selected (i.e., when it crosses a sector boundary) the radio button is no longer highlighted. This combination of no change to the target's on-screen icon and persistence of the classification in the information window is the Dot interface.

The third study (AP3) manipulated the ease of retrieving history information (i.e., "have I classified this target already") from the display. In addition to Dot interface, two new interfaces were used. The No-dot interface does not display the classification status of the target in the radio button. The color interface changes the color of the on-screen target icon when the target is classified. When a target is no longer classified (i.e., when it crosses a sector boundary) the icon reverts to the unclassified color.

In the first two studies, subjects frequently reselected already classified targets. For example, in the course of one 15-min scenario, one notable subject reselected 225 already

classified targets; however, none of these were reclassified (i.e., the subject selected the target by clicking on its icon, but after looking at the information window and seeing that one of the seven threat value radio buttons was "on," he found and clicked on another target icon). Although this subject reselected twice as many targets as the average subject, his behavior was representative. There was no indication that subjects reselected targets with the goal of reclassifying them; rather their persistent selection of already classified targets suggests that for the Dot interface subjects do not remember whether a target has been classified. The time spent checking already classified targets results in less time spent on unclassified ones.

It was unclear in the AP1 and AP2 studies whether this memoryless strategy (Ballard, Hayhoe et al. 1995) is adopted by choice or whether, under the conditions of the study, human cognition is incapable of retrieving target status information. AP3 tested empirically the memoryless strategy issue by providing three interfaces, each intended to force a different amount of cognitive effort in terms of memory retrievals. AP3 was successful in showing that changes in the interface will enable different sets of strategies that are differentially successful and result in large differences in performance. Twelve different scenarios were created for this study. Although each scenario had to be different, an attempt was made to equate the difficulty of each. Over the entire study (and including practice), each subject performed 12 scenarios. The order in which the scenarios were presented was partially counterbalanced.

In AP4 the color interface was eliminated because the results of AP3 showed that the access cost (i.e. time to access the information about the target) of this interface was so low, that the memoryless strategy was always adopted. A new interface was added in AP4 to provide a clearer indication of when subjects were using immediate feedback on the accuracy of their decision. The immediate feedback interface consists of a new window devoted to providing feedback. The window however is initially covered with a gray window when it is displayed. The

subject must move the cursor over the gray window and click on it to uncover the feedback indicator. The window is recovered when the mouse leaves the window. Thus to access the information in the environment there is a perceptual-motor cost.

The AP4 experiment was conducted over a three hour session. In the first two hours, the subjects were instructed on how to do the task, did a sample scenario with the experimenter, did one 9-minute practice scenario, and then did four 12-minute scenarios in each of the four conditions. In the third hour of the session the subjects did four more 12-minute scenarios in each of the four conditions. The order of the conditions was counter-balanced over a block of 8 subjects. The subjects were eyetracked with an ASL E5000 eyetracker. Eight of the twelve scenarios developed for AP3 were used in AP4. Twenty-four undergraduate students were run and received course credit or $50 if they completed all the scenarios. The data from these twenty-four subjects is the human data used to validate the Argus model and will be presented in the results section.

The fifth Argus Prime experiment (AP5) was intended to increase workload by adding a secondary task to the primary target classification task. A tracking task was chosen to increase the perceptual and motor activities, but not the memory processes. The classification task was identical to AP4. The main research question was what would be the effect of increasing perceptual and motor requirements on a task that required cognitive effort as well as perceptual and motor effort.

The AP5 experiment was conducted over two sessions, lasting two and three hours. The first session was two hours. During this session the subjects were instructed on how to do the classification task, did a sample scenario with the experimenter, did one 9-minute practice scenario, and then did four 12-minute scenarios in each of the four conditions. The subjects were not eyetracked. In the first hour of the second session, the subjects reviewed the classification

task and were trained in the tracking task. In the last two hours, the subjects did two blocks of four 12-minute scenarios. For each scenario, the subject had to perform the classification task and the tracking task. The subjects were instructed to try to do equally well on both tasks. Each block had one scenario in each condition. The subjects were eyetracked with an ASL E5000 eyetracker. Twenty-four subjects were run in AP5 and received course credit. The data from these twenty-four subjects is the human data predicted by the Argus model and will be presented in the results section.

4.1. Research Method

The research paradigm for this dissertation, shown in Figure 7, compares the results of two kinds of efforts. On one side a cognitive architecture, which provides a theory of embodied cognition, is combined with task knowledge to develop a process model. The process model makes predictions about human performance.

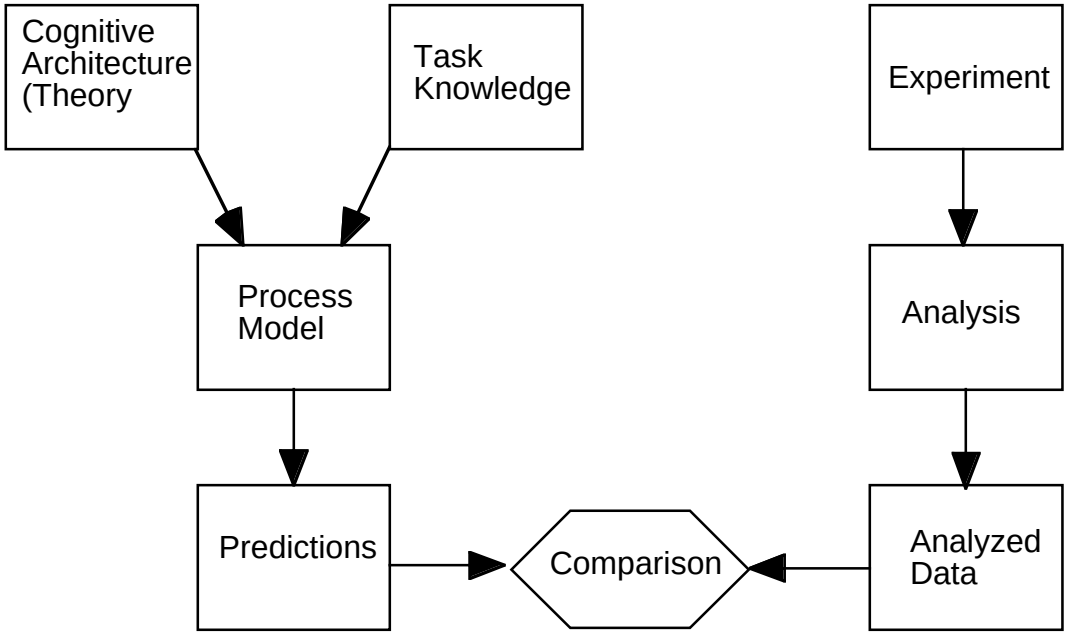


Figure 7. Research paradigm for cognitive modeling. Adapted from (Taatgen 1999).

For the second input to this comparison, experiments are conducted with human participants performing the task, experimental data is collected and analyzed. The predictions of the models are compared with the analyzed human data. This paradigm directly tests predictions of the model and indirectly tests the architecture. Differences between the model predictions and analyzed human data can be due to errors in the model, the task knowledge or the architecture.

5. The Argus Model

The Argus model produces its predictions about interactive behavior by the interaction of task knowledge, the nature of the environment, the current task state, and acting on the environment. Task knowledge is encoded in procedural memory as productions. The task state is contained in declarative memory. The environment is sensed through the perceptual module and acted on by the motor module. The detail of how these mechanisms produce interactive behavior that emulates human behavior is expanded in this chapter.

5.1. Overview

As previously stated, the human participants were given one hour of training and practiced on four scenarios. Their performance improved on each scenario. The Argus model makes the assumption that after the training and practice the knowledge to perform the task has been compiled from task instructions to procedures and therefore can be modeled by productions. The goal structure of the Argus model mirrors the task decomposition. In the single task environment, which will be called the classification task, three main unit task goals control the execution of the task as seen in Figure 8. In the dual task environment an additional unit task, called the tracking task, is interleaved with the classification unit tasks by the addition of the tracking unit task goal as seen in Figure 9.

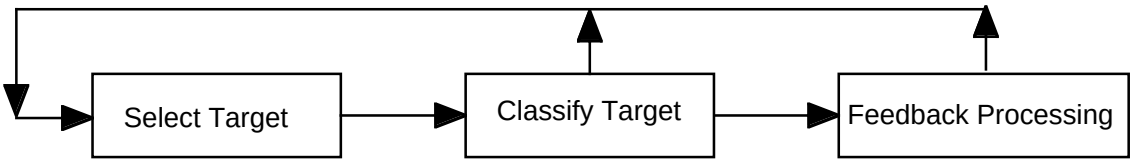


Figure 8. Goal flow for the single task environment - AP4.

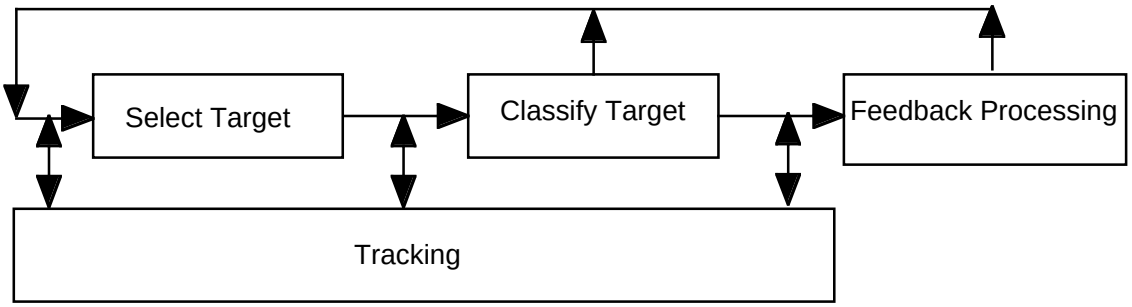


Figure 9. Goal flow for the dual task environment – AP5.

Before the Argus model runs, a well-learned (i.e. a high base-level activation) chunk for each unit task goal is created in declarative memory. To switch between the classification task goals, the goal of the next task is retrieved from memory. This reflects the assumption that the classification task is performed routinely at the unit task level. Each of the unit task goals spawns other goals to accomplish the unit task. One reason for this subgoaling is to limit the number of slots in each chunk to be within the limit set by the ACT-R theory. Another reason is that it is below the unit task level where subject strategies emerge. When the tracking task is being performed a goal is created rather than being retrieved. This is due to the episodic nature of the tracking task and the fact that it is triggered by interface conditions rather than routine behavior.

The Argus model is sensitive to the experimental conditions. It assumes that when the scenario starts, subjects know which experimental condition is in effect for this scenario. The dot vs. no-dot condition is implemented in the Argus model by lowering the utility of the productions that look at the dot when a target is selected. The feedback vs. no-feedback (fb vs. no-fb) condition is implemented by lowering (in the no-fb condition) the utility of the productions that detect the presence of the feedback window.

The variability of the human subject data is a strong indicator that different combinations of actions (microstrategies) do occur. The Argus model tries to capture some of these microstrategies. The fact that the Argus model does not simulate all the possible combinations is an indication that the Argus model will not achieve a perfect fit to the human data. The interleaving of memory, perceptual and motor actions is an important consideration in capturing the combinatorial nature of interactive behavior. The Argus model attempts to capture the variability through the matching and conflict resolution processes of the architecture. If an action has not been performed then a production to implement that action will match and it will enter the conflict resolution processes. The productions that do match and enter the conflict resolution and may have the same utility (since the Argus model does not enable sub-symbolic production learning), but noise is added to each computation of utility so that which one is chosen will vary, thus adding variability to the execution order.

The number of combinations of these operations is also influenced by the experimental condition. In the dot condition, the interface design makes the classification state of the target available in the environment. In the dot condition, moving and clicking on the target may be preferred to memory retrieval of target state. In the no-dot condition, more effort may be invested in rehearsing the target classification encoding. The Argus model implements this sensitivity to

interface conditions by setting the utility of the relevant productions based on experimental condition.

The processing of each unit task will be discussed in later sections.

5.2. Model Structure

5.2.1. Architecture Parameters

As described earlier, the ACT-R 5.0 architecture defines a number of parameters that can either be left at their default values or set by the model. The settings are contained in the code and do not change from run to run. This section describes the parameter settings that are relevant to the Argus model and are summarized in Table 5.

Table 5. ACT-R 5.0 parameter settings for relevant parameters.

Type	**Name**	**Value Used**	**Recommended or Default**
Rational Analysis	Enable Sub-symbolic Computation	T	T
Rational Analysis	Expected Gain Noise(S)	0.20	None
Activation	Base-level Learning	0.50	0.50
Activation	Activation Noise	0.25	0.30
Activation	Latency Factor	0.05	None for ACT-R 5.0

The Argus model enables sub-symbolic computations by setting the Enable Sub-symbolic Computations parameter to the default (T-true). This directs the architecture to enable the Bayesian computations for conflict resolution and declarative memory activation.

The Expected Gain Noise parameter specifies the *S*-value of the Gaussian noise added to the *PG-C* evaluation of utility used in conflict resolution. The Argus model sets this parameter to 0.20. This parameter does not have a recommended setting.

The Base Level Learning parameter enables learning of the base level activation of declarative memory chunks. The model enables this learning mechanism with the default value. The setting has been used in many ACT-R models that are concerned with sub-symbolic learning and is the recommended value in the ACT-R manual (Lebiere 1998).

The Activation Noise parameter specifies the *S*-value of the Gaussian noise added to each chunk activation at every cycle. The Argus model sets this parameter to 0.25 which is close to the recommended value.

The Latency Factor parameter is a scaling factor for the retrieval time equation. ACT-R 5.0 uses a new retrieval time equation and no recommended setting has been determined.

The associative link mechanism and partial matching are not turned on. These are the default settings for ACT-R 5.0 at the current time.

5.2.2. Declarative memory

The Argus model assumes that the subjects are trained on performing the task. Their task knowledge is represented in the productions that carry out the task and in declarative memory elements that represent the goal structure at the unit task level. These goals are well learned and control the execution of the task as seen in Figures 8 and 9. The declarative memory elements for these goals are added to declarative memory prior to model execution.

The human subjects were taught the classification algorithm. It is assumed that during the training process declarative memory elements representing the mapping from raw value to cue value and memory elements representing the weight of each attribute were created in declarative memory. One source of human variability is that the level of learning these elements varies from human to human. In ACT-R terms the amount of base level activation is different in each subject. The mapping and weight elements are created in the Argus model as part of memory initialization. The mapping element type specification is

(chunk-type real->cue attribute cue-value low high), where
attribute is range, altitude, or approach distance,
cue-value is a memory element representing integers 0 – 3,
low is the smallest numerical raw data value for the specified cue value, and
high is the largest numerical raw data value for the specified cue value.

The weight memory element type specification is

(chunk-type weight attribute wgt), where
attribute is range, altitude, or approach distance, and
wgt is a memory element representing integers 0 – 2.

5.2.3. Procedural memory

The Argus model consists of 243 productions. Figure 10 shows the breakdown of the productions by function.

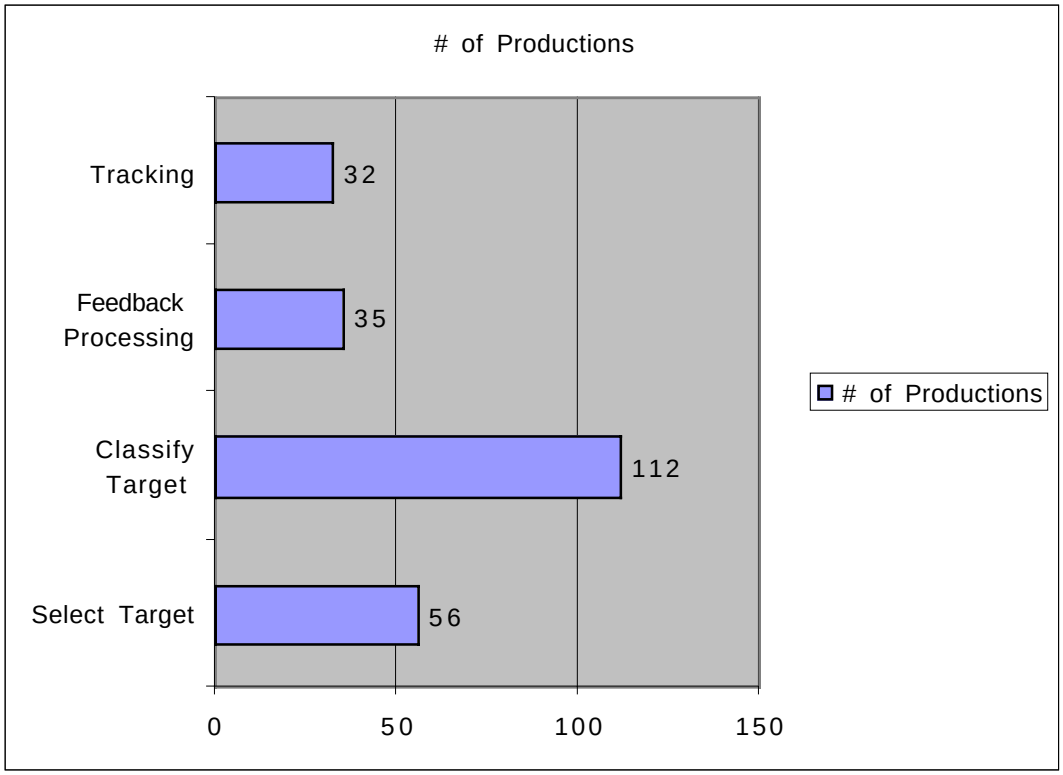


Figure 10. Model breakdown by unit task goal type.

Interactive behavior in the Argus model is possible because the ACT-R 5.0 architecture has “eyes” and “hands”. The environment is perceived and acted upon by performing buffer operations. The breakdown of productions by buffer actions is shown in Figure 11.

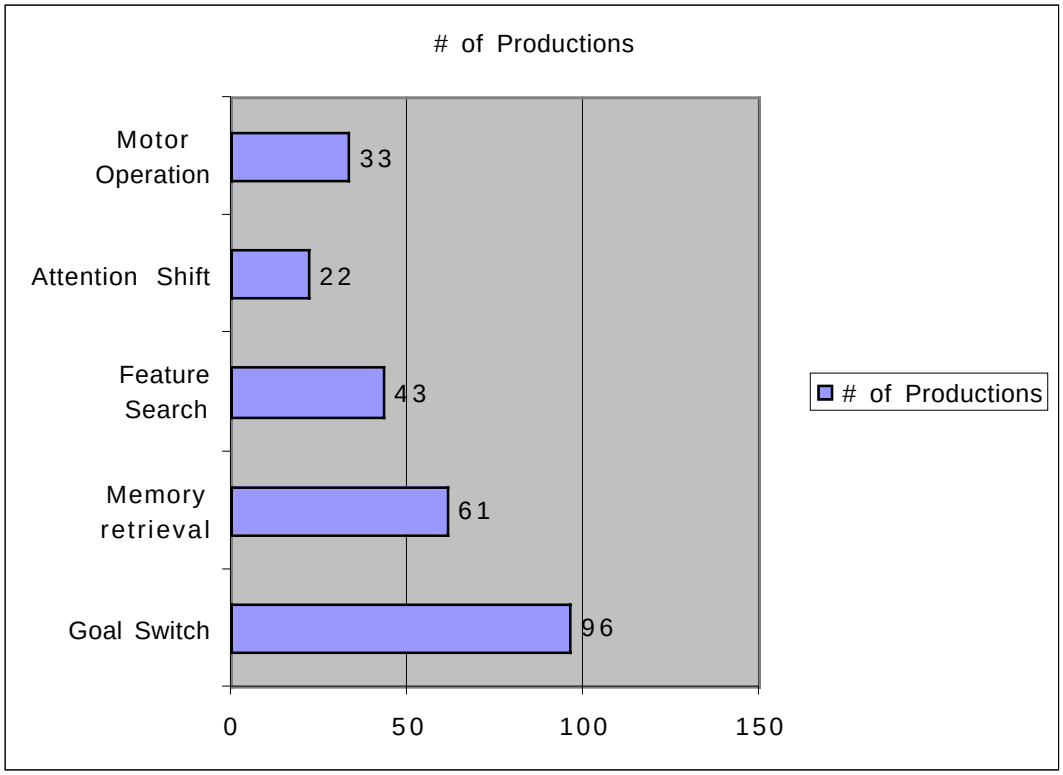


Figure 11. Model breakdown by function.

5.3. Simulated Human Users

To account for between-subject variability the Argus model provides a mechanism for the cognitive or design engineer to build a simulated human user. A simulated human user is a particular instantiation of the model. A particular simulated human user is created by the engineer using the dialog shown in Figure 12. When Done is pressed the settings are saved to a user definition file. This file is similar to that of a human user and establishes the simulated human user as a user of the system. The fact that the user is a simulated one rather than an actual human is transparent to the Argus software.

In most interactive environments, the computer interface contains features that are intended by the designer to aid the user in performing the task. What is difficult for the designer to control is the actual use of the feature. Humans exhibit a range of behaviors in actual use of a feature. The interface characteristics section contains slider bars on a number of interface features to achieve a mix of behaviors. The simulated human user file contains information which influences model execution. Simulated human users are created in order to mirror the mix of strategies found in the human subjects. For a computational model to successfully simulate a human in a highly interactive environment, it is crucial that the mixture of cognitive, perceptual and motor operations in the model closely follow that of the human users being emulated. The creation of simulated human users is an approach toward this goal.

Slide bars are used to provide a range of possible settings for each parameter since subject behavior is a mix of strategies. That is, some subjects will always use an interface feature, some subjects will never use the feature, and other subjects use the feature some of the time.

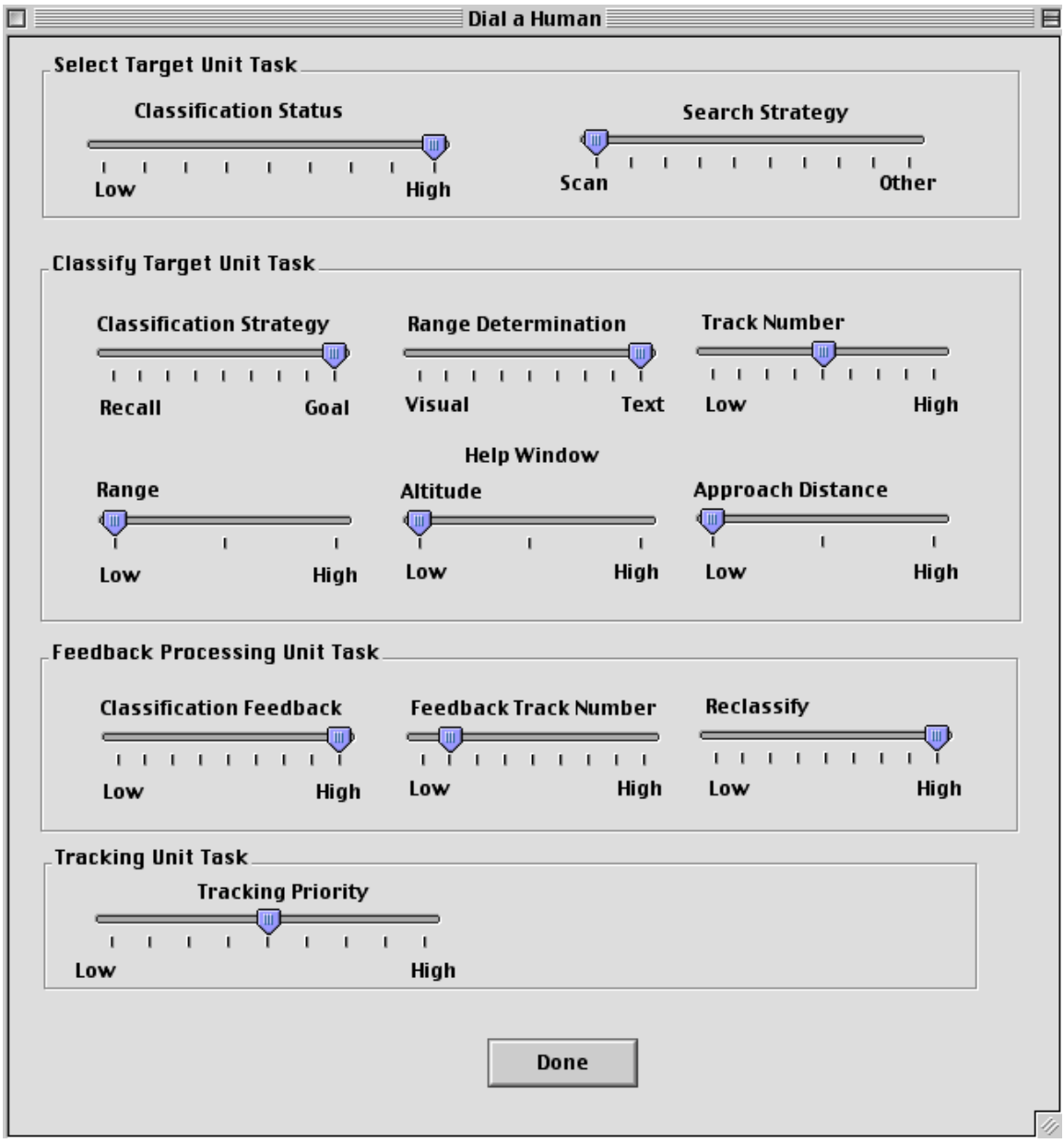


Figure 12. Dialog Window used to configure a simulated human user.

Between-subject variability in interactive behavior arises when the task or the interface permits multiple actions to accomplish a task goal or the order of the actions is not fixed. Given the opportunity, different humans will adopt different strategies to perform a task or will use a

different interface feature. At the unit task level, humans may be aware of the strategies they follow, but at lower levels they generally are not aware of these strategies (microstrategies in the Gray and Boehm-Davis sense). A significant problem in user interface design is determining what strategies users will adapt given the set of features designed into the interface, and how these different strategies will affect performance.

Another source of between-subject variability is that different people have different cognitive capacities (Just and Carpenter 1992). Cognitive capacity may influence the use of perception and motor processing. For example, people with low working memory capacity may be more likely to use perception to access information in the world.

The simulated human user configuration screen provides a range of settings for many of the choice points in the Argus tasks and interface. Each parameter has an effect on the mix of cognitive, perceptual, and motor operations.

As seen in Figure 12, the dialog is organized by unit task boxes. Each box groups the parameters for one unit task.

5.3.1. Select Target Unit Task Group

The Select Target unit task can be decomposed into a visual search task, attending and clicking on the target, and target status determination (is the target already classified?). Analysis of eye data in the third Argus experiment indicated visual search strategies as a rich source of between-subject variability. The set of strategies implemented in the Argus model is discussed later on. The configuration dialog provides two parameters, Classification Status and Search Strategy, to account for subject differences in target selection.

The Classification Status slider determines how often the SHU uses the filled in radio button (the “dot”) to determine if a target is classified. The dot represents information stored in the world. The alternative is to use information stored in the head. Subjects vary as to whether

they try to retrieve the fact the target has been classified or whether they use perceptual and motor functions to access the environment. The scale for this slider is a probability scale. The slider sets the probability that the SHU will access the dot by moving the cursor to the target, selecting it to bring up the information window, and looking for the dot (perceptual-motor access) versus trying to remember if the target has been classified (memory access). The probability of perceptual-motor access increases from left (Low setting) to right (High setting).

The data analysis showed that most subjects use a systematic scan strategy at the beginning of the scenario. This initial systematic scan strategy lasts for a varying amount of time. The Search Strategy parameter controls the length of time from the beginning of the scenario in which the probability of the scan strategy being executed is higher than the other strategies. The left hand setting (Scan) sets the time at a maximum. The right hand setting (Other) sets it to a minimum value. There certainly are more points of strategy differences in visual search in this task and identifying them is a topic for future work.

5.3.2. Classify Target Unit Task Group

The settings in the Classify Target unit task group control how the SHU computes the threat value of a target. The threat value of a target is often incorrectly calculated. One source of error in the calculation procedure is incorrectly remembering results calculated at intermediate stages of the algorithm. For example, the target score is the sum of the cue values for each of the attributes. When this sum is calculated the weighted cue value of each attribute, which previously had been calculated and stored, must be recalled. The possibility of recalling the wrong intermediate result is a function of how the result was stored in memory. If the result was stored as part of the goal the result will be directly available. If not, then the result must be retrieved and is subject to error. The classification strategy slider bar setting influences the storage strategy of

the intermediate weighted cue value results. The scale is a probability scale. The probability of being able to directly retrieve the result increase from left (Recall) to right (Goal).

The Range cue value can be determined in two different ways. The first way is that the cue value for range can be visually determined from the targets position on the radar screen since each sector corresponds to a cue value. For example, the inner sector extends from 0 to 50 miles and the raw data value range for a cue value of 3 is also 0 to 50 miles. The second way is by reading the target's range, displayed as text in the information window, and mapping the data value to the proper cue value. The Range Determination slider influences how the range cue value is determined. The scale is a probability scale. The probability that the range cue value will be determined from text increases from left (Visual) to right (Text).

Remembering whether a target has been classified is an important part of the overall task. The track number of the target being classified is displayed in the information window to provide a visual reminder. Attending to the track number will strengthen the episodic memory of classifying this target; however, the actual classification does not require the user to use this feature. The Track Number slider determines how often the SHU looks at this track number. The scale is a probability scale with the probability of attending to the track number increasing from left (Low) to right (High).

Help in calculating the weighted cue value for each relevant attribute is accessed by clicking on the attribute (e.g. Range) label. This displays the help window. The user must then retrieve the raw value since the info window is covered by the help window and determine to which range of displayed values it falls within, and then find and encode the cue value. To get back to the info window the Done button is clicked. The point is that help is accurate since the needed information is in the world but is quite expensive in terms of time and perceptual motor effort. This again is a source of between-subject variability. The Argus model will always use

help if it can't retrieve the learned mapping chunk. The three slider bars for help partially control the model's use of help by specifying the degree to which the mapping chunks are learned. In the lowest setting, help is never used since the mapping chunks are well learned. In the middle setting, the mapping chunks start out not being well learned but are strengthened each time help is used. In the highest setting, help is always used since the mapping chunks are not well learned and no additional learning takes place during help processing.

5.3.3. Feedback Processing Unit Task Group

Feedback processing consists of uncovering the feedback by clicking on the gray window cover, noticing the color of the feedback icon, and deciding to reclassify if a classification error is indicated. Users vary in their use of the feedback window since it requires perceptual and manual effort to access the feedback window plus cognitive effort to process the feedback. The Classification Feedback slider determines how often the SHU accesses the feedback in the feedback condition. The scale again is a probability scale. The probability of the SHU accessing the feedback increases from left (Low) to right (High).

If the target was incorrectly classified, the track number (needed to find this target on the radar screen) is provided in a separate dialog box. The track number is covered by a gray cover so the user must perform a manual action (click on the gray cover) to see the track number. When the feedback indicates a correct classification, the track number may be used to encode the fact that this target has been classified. The Feedback Track Number slider indicates how often the SHU accesses the track number. The scale is a probability scale. The probability that the SHU looks at the track number increases from left (Low) to right (High).

User also vary on how often they immediately reclassify a target when an error is indicated. If the feedback window is accessed and the feedback indicates an incorrect classification, the user may choose to ignore the error because of the effort required to search for

the target. The Reclassify slider determines the probability that the SHU will attempt to reclassify a target if a classification error is indicated. The scale is a probability scale. The probability of reclassifying an incorrectly classified target increase from left (Low) to right (High).

5.3.4. Tracking Unit Task Group

Users vary in the amount of time they spend doing the tracking task, even though, in the dual task environment, the tracking task and classification task are of equal importance. To aid the user in balancing performance, performance scores for each task are displayed in the status bar at the top of the screen. A large difference in score indicates that the task with the lower score requires more effort. The Tracking Priority slider setting determines how often the SHU will switch from the classification task to the tracking task and how long the user will stay in the tracking task. The number of switches and the time spent in the task increases from left (Low) to right (High).

5.3.5. Actual Model Settings

Table 6 specifies the human performance measure used in estimating the initial setting of each slider bar in AP4.

Table 6. Performance measures for estimating slider bar setting for a SHU.

Slider Bar	**Human data**
Classification Status	Mean number of targets selected during a scenario (includes already classified targets).
Search Strategy	None.
Classification Strategy	Mean number of targets correctly classified during a scenario (includes reclassifications).

Range Strategy	None.
Track Number	Mean number of targets selected during a scenario(includes already classified targets).
Help Window - Range	Mean number of times that the help window was accessed for range during a scenario.
Help Window - Altitude	Mean number of times that the help window was accessed for altitude during a scenario.
Help Window - Appdist	Mean number of times that the help window was accessed for appdist during a scenario.
Classification Feedback	Mean number of times the feedback window was uncovered during a scenario.
Feedback Track Number	Mean number of times feedback track number uncovered during a scenario.
Reclassify	Mean number of target reclassifications immediately after accessing the feedback widow during a scenario.

The parameters were changed in order for the Argus model to fit the data results of AP4. The same parameters were then used in AP5 with the addition of the tracking parameter which was estimated from the tracking data. Table 7 lists the actual model settings used in both AP4 and AP5.

Table 7 Model-subject settings used in both AP4 and AP5.

SHU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Classification Status	9	8	8	7	7	6	6	5	5	4	4	9	9	9	5	5	9	9	9	9	9	9	9	9
Search Strategy	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	6	6	6	6	4	4	4	4
Classification Strategy	9	1	9	9	9	5	5	5	9	9	9	9	5	9	9	1	9	9	9	9	9	9	7	7
Range Determination	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Track Number	1	5	5	5	5	5	5	5	5	9	5	5	9	9	9	9	1	1	1	1	1	1	3	7
Help Range	0	0	2	1	0	0	2	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Help Altitude	0	1	2	0	0	0	2	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Help Appdist	0	1	2	1	0	0	2	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Classification Feedback	9	9	9	9	9	5	9	1	9	5	9	9	1	9	5	9	9	9	9	9	9	1	3	9
Feedback Track Number	1	5	5	5	1	1	1	1	9	1	1	1	9	9	5	5	1	1	1	1	1	1	5	1
Reclassify	1	9	8	5	5	9	9	1	9	1	9	9	9	9	1	5	6	6	6	4	9	1	9	5
Tracking	2	2	2	8	5	8	4	2	3	2	3	2	6	9	2	5	7	6	2	2	3	4	2	7

5.4. Argus Unit Tasks

As stated above, the nature of the Classification task imposes a linear order on its three unit tasks: Select Target, Classify Target and Feedback Processing. This section describes the model's operation by presenting a high level production flow chart for each unit task. The circles represent conflict resolution points (i.e. decision points). The arrows extend from a circle point to the productions in the conflict set. The conflict set contains all those productions which match the current state of the ACT-R 5.0 buffers. Thus, the arrows indicate which production are eligible to execute (i.e. possible execution paths). The production actually chosen (and therefore the one that executes) varies depending on the experimental condition, which strategy is being employed by the simulated human user, and state of declarative memory. Lines with arrows at both ends

indicate that after the production (or series of productions) executes the Argus model returns to the conflict resolution point and the same production or a different production could execute.

5.4.1. Select Target

Select Target consists of locating a target on the radar screen, moving visual attention to the target, determining the classification status of the target, encoding the target, moving the cursor to the target, and clicking on the target. In those experimental conditions where the classification indicator is provided as an interface design feature, the checking of the indicator is part of target selection. Target selection continues until a target is found that requires classification (at least the human subject or the SHU "thinks" it requires classification). While interface and task constraints do impose some order on the sequence of steps, analysis of the human data has shown that these steps are not done in strict order.

The production flow for the Select Target unit task is shown in Figure 13.

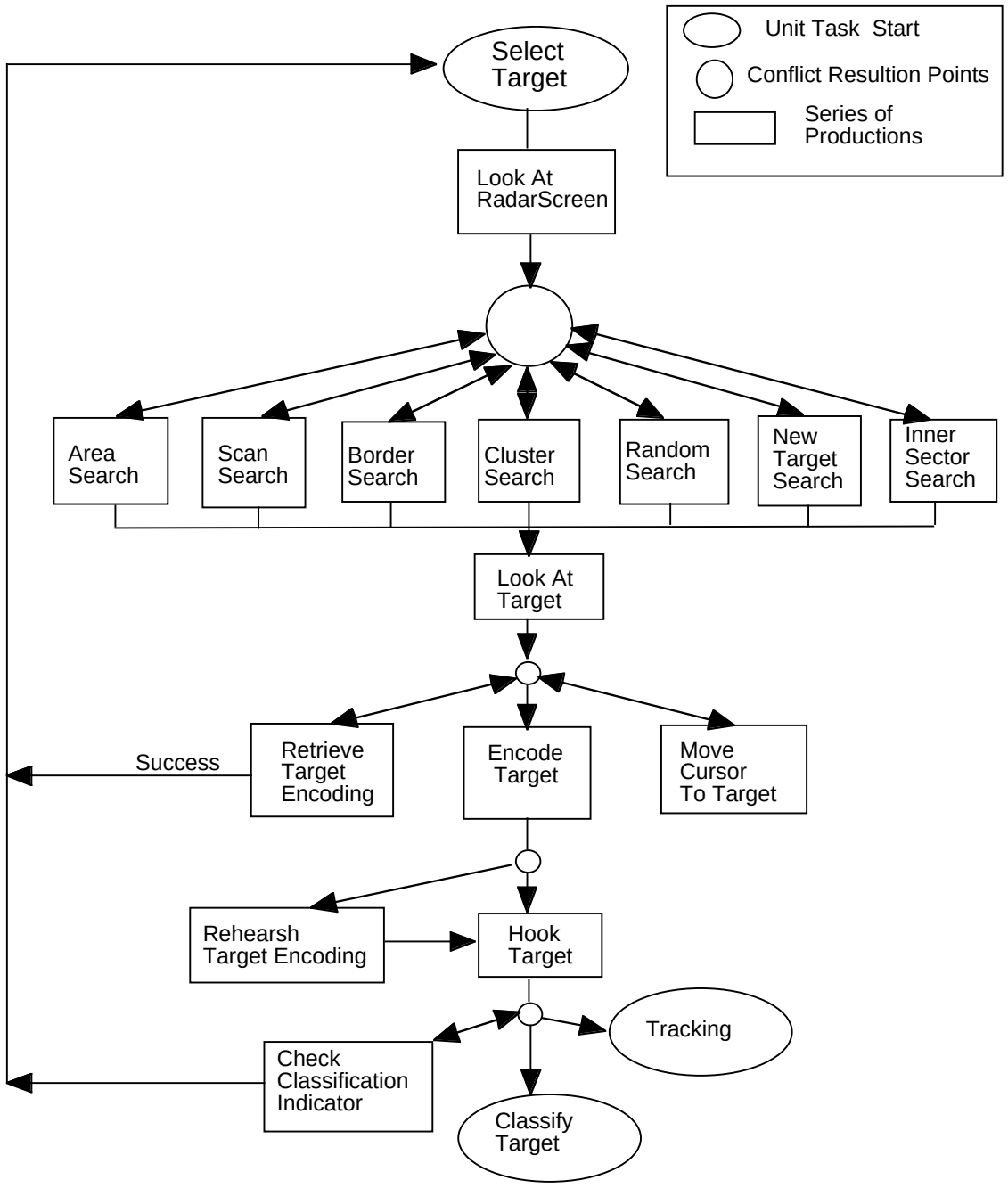


Figure 13. Flow of Select Target Unit Task.

The unit task begins with recalling the location of the last target selected and moving attention to that location. The Argus model must then choose a search heuristics. The heuristics coded into the Argus model are ones that users employ. During the training scenarios the users build up a toolbox of these heuristics. Which heuristic is actually employed at any one time is a function of the current screen configuration (i.e. the spatial distribution of targets on the screen), the task constraint of classifying each target in every sector, and the cognitive capabilities of the user. By playing back the log files of users doing the task, nine heuristics have been identified. These heuristics have been implemented in the Argus model as feature searches using the vision module feature search mechanism.

The problem of which heuristic to choose is difficult to implement in ACT-R. The problem is two-fold. First, ACT-R does not have a scene analysis capability. The vision module returns only one location that meets the search criteria. If more than one location met the search criteria then the module randomly picks one of those locations and places it in the visual location buffer. The second problem is in the production selection process. Conflict resolution is based on the Expected Gain (Utility) of each production being considered at the sub-symbolic level. The Expected Gain equation is based on the history of the production in a certain context. The problem of applying it to heuristic selection in this case is that the environment is dynamic. The configuration of targets on the screen is constantly changing. The Argus model solves this problem by extending the ACT-R architecture with a function that replaces the normal ACT-R conflict resolution (represented by the large circle in Figure 13). This function selects a target location strategy that it computes to have the highest probability of success. This function first builds a the current context which includes:

- Number of targets approaching border
- Number of targets just crossed a border

- Number of targets within 1 and 2 degrees of visual angle centered at current location
- Number of new targets
- Number of targets in current segment
- Number of targets near own-ship
- History of success or failure
- Scenario time

For each heuristic, a probability of success and a PG-C with noise is calculated based on the current context. The heuristics are ranked according to their utility and the highest one selected.

For each target location strategy, one initial production with a goal specification of select-target and step equal to find-target is defined. All of these initial productions enter the conflict set. At this point the normal conflict set resolution is suspended and the conflict resolution method described above is executed. When the initial production that is chosen fires, it switches the goal to match the specific goal type of strategy it is initiating. The search strategies that are implemented in the Argus model are shown in Table 8 with the feature that is used to constrain the search. All heuristics specify the kind feature as a target icon and the attentional marker as nil. The productions that actually implement the strategy then execute.

To locate a target, a feature search is executed using the ACT-R 5.0 feature search mechanism. This mechanism specifies a set of features which can be detected. A search strategy is based on a different combinations of features that constrain the search. A number of different strategies have been identified by playing back the human eye data. All the search strategies are eligible for selection during conflict resolution.

Table 8. Description of target search strategies.

Heuristic	**Feature(s)**	**Description**
Within an area	Screen-x (within x1 x2) Screen-y(within y1 y2) Value segment #	Finds a target icon in the rectangle specified by (x1,y1) (x2,y2) and is in the segment specified by segment #.
Scan right to left	Screen-x less-than –current Value segment # Nearest current	Finds the nearest target icon that is to the left of the current focus of visual attention and is in the segment specified by segment #.
Cluster	Nearest current	Finds the target icon that is nearest the current focus of visual attention.
Cluster-in-seg	Value segment # Nearest current	Finds the target icon that is nearest the current focus of visual attention in the segment specified by segment #.
Border	Kind arc Nearest current	Finds the border nearest the current focus of visual attention.
	Arc # (implementation note: color slot used for arc) Nearest current	Finds the target that is nearest to focus of visual attention and within 15 miles of border.
Border-by-seg	Kind arc Value arc #	Finds the border specified by arc #.
	Arc # (implementation note: color slot used for arc) Nearest current	Finds the target that is nearest to focus of visual attention and within 15 miles of border.
New-target	Attended new	Finds a target with the new marker set.

Random-in-seg	Value segment#	Picks a target at random within the segment specified by segment #.
Random-in-current-seg	Value current-segment #	Picks a target at random within the current segment.
Random	None	Picks a target at random.

At the completion of the feature search, if the feature search is successful a visual-location declarative memory element is created and placed in the visual location buffer. Control is switched back to the select-target goal, with the step slot set to attend-to-target. If the feature search is not successful, then control is switched back to the select-target goal, with the step slot set to select-new-strategy. The current strategy is discounted so that it has less probability of being selected again.

At this point, a mix of actions could take place. The possible actions are: retrieve target encoding, encode target, and move cursor to target (see Figure 13). One or more of the actions may execute (possibly in different orders). If a retrieval is attempted and is successful the Argus model goes to the top of target selection. To encode the target (encode target box in Figure 13), attend-to-target production issues a Move-Attention to the Vision Module through the visual buffer, using the location stored in the visual location buffer. The Vision Module moves the eye and visual attention to the location and adds to declarative memory a representation of the screen object. Moving visual attention allows the visual system to encode properties of the object itself (vs. features at a visual location). The representation created by the vision module is an episodic trace containing the screen location, the text of the object (i.e. the track number), the screen segment it is in and its relationship to the nearest border.

If the target was encoded, then a combination of rehearsing this fact or hooking the target can occur. The final step of target selection is a combination of checking the classification indicator and initiating the Classify Target unit task.

5.4.2. Classify Target Unit Task

The Classify Target unit task consists of processing target data in the information window to calculate a target score and then mapping that target score to a 1 – 7 threat value for that target and entering the decision into the system. The variability in this unit task, which the Argus model attempts to capture, comes from the interface design and the embodied cognition components of the ETA triad.

The Classify Target unit task goal is specified as

(chunk-type classify-target step tn-sg cv-sg ts-sg calc-threat-value-sg), where

step provides control flow,
tn-sg is a sub-goal for verifying the track number,
cv-sg is a sub-goal for calculating a weighted cue value,
ts-sg is a sub-goal for calculating the target score, and
calc-threat-value-sg is a sub-goal for determining the threat value of the target.

A classify-target memory element is added to declarative memory during initialization. The base level activation of this element is set very high, based on the assumption that the subjects have been trained and the goal structure has been well learned. The high base level activation ensures it can be successfully retrieved.

Figure 14 depicts the overall flow of the Classify Target unit task.

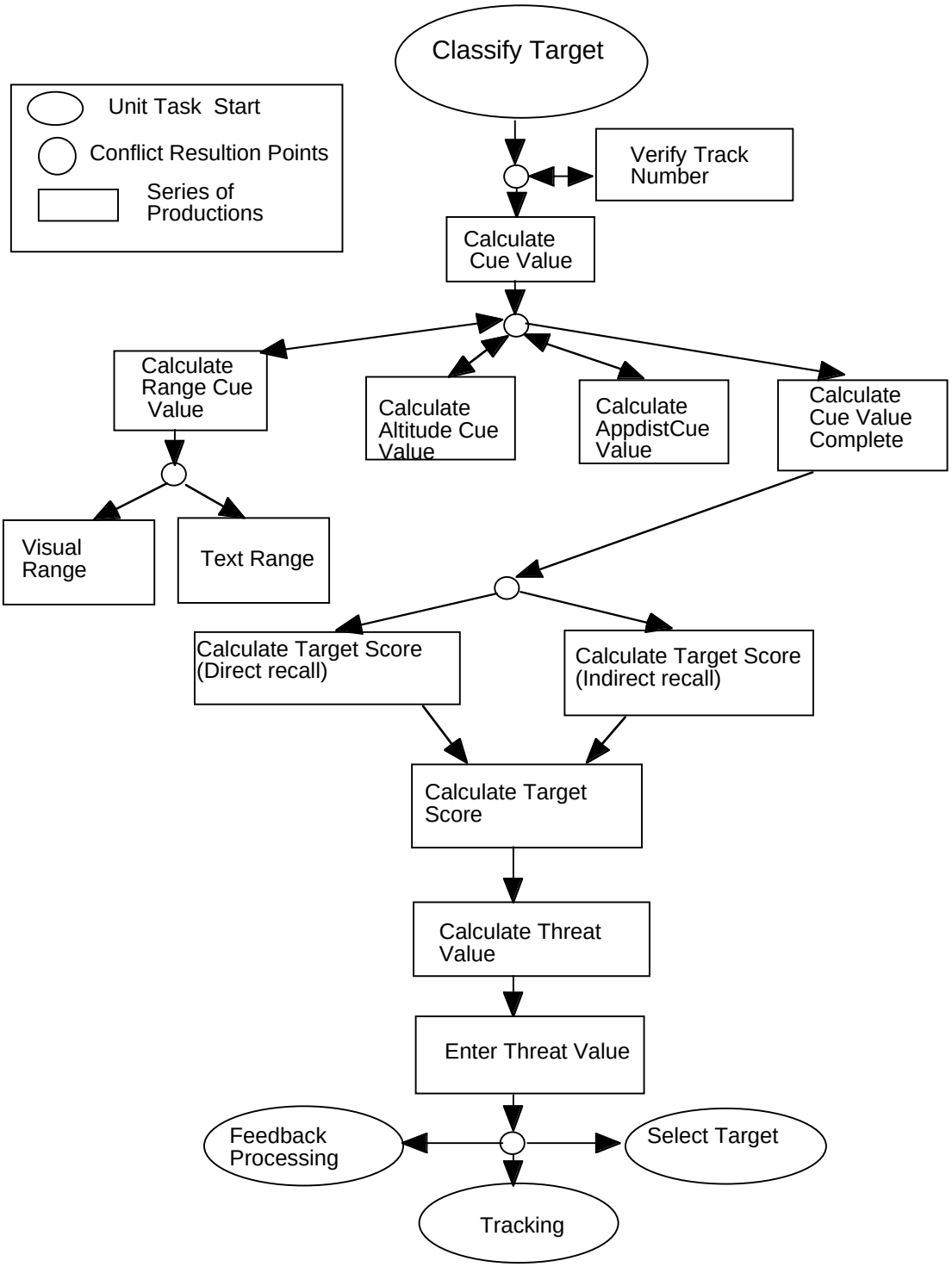


Figure 14. Flow of the Classify Target unit task.

5.4.2.1. Verify Track Number

The information window contains the track number of the currently selected target. The Argus model chooses between looking at the track number or not looking at the track number. If the former is chosen then the verify track number sub-goal is executed. The sub-goal serves two purposes. It verifies that the information is for the intended target and it strengthens the encoded representation of the target. A sub-goal is set to the read the track number in the information window. When the track number is read it is the focus of visual attention. A memory retrieval for the target specified by the track number contained in the visual buffer is executed. Thus information in the environment is providing a cue for the memory retrieval. Successful retrieval strengthens the target encoding. If the retrieval fails, this indicates that the target clicked on was not the intended target (e.g. two targets could have been close together and the mouse click was actually on a target other than what the subject intended).

5.4.2.2. Calculate Cue Value.

The target score in AP4 and AP5 is based on range, altitude, and approach distance. The cue values for these attributes are calculated in random order. The randomness is obtained from the conflict resolution process. Initially, three productions, each initiating the calculation of one of the three attributes, complete for execution (see Figure 14). They all have the same utility but a different amount of noise is added to each one, so one will win and execute. When it completes, it will no longer be added to the conflict set, so the remaining two compete. Finally, only one is in the conflict set, so it executes. Calculating the cue value is a two step process. First, the raw score is mapped to a integer value (0 – 3), then this value is multiplied by the weight given to the attribute. This process is described in detail below.

For each attribute, the sub-goal to read the corresponding raw data text in the information window is executed and it places the text in the visual buffer. The text is converted to a number. The attribute name and numerical raw data value are used to retrieve a real->cue declarative memory element. The retrieval code is

```
+retrieval>
        isa             real->cue
        attribute       =attr           ;;; attribute name
        cue-value       =cv             ;;; cue value will be here
<=      low             =raw-value      ;;;; low end of range
>=      high            =raw-value      ;;;; high end
```

The ability to match using greater-than and less-than operators is new to ACT-R 5.0. While the mechanism reduces the probability of a retrieval error due to mismatches, the retrieval can still fail if the real-cue memory element does not have sufficient activation to be retrieved. If the retrieval does fail then the Argus model sets a goal to perform help processing (i.e. accessing the help screen and calculating the cue value from the explicit information on the help screen).

The next step is to weight the cue value. The declarative memory for the weight given to the attribute is retrieved. If this retrieval fails, help is accessed to obtain the weight. The multiplication fact for multiplying the weight times the cue value is retrieved. Multiplication facts are well learned so there is a very low probability of a retrieval failure. A weighted cue value memory element for the attribute being calculated is created from the result of the multiplication. This chunk is rehearsed based on the rehearsal paradigm developed by Altmann (Altmann and Gray 2000). It is rehearsed more in the no-feedback condition than in the feedback condition. The user data indicates that in the no-feedback condition users spend more time in the Classify Target unit task. The assumption is that since the user will not be able to determine accuracy from the feedback, this extra time is spent in memory rehearsal in order to improve accuracy. A new goal is set to calculate the target score from the calculated cue values.

5.4.2.3. Calculate Target Score

Target score is the sum of the three weighted cue values. To add three numbers, two of the numbers must be added to form an intermediate result, the third number is then added to the intermediate result. To execute this algorithm in ACT-R 5.0 a number of memory retrievals are involved. The Argus model implements two strategies with different error characteristics for the retrieval of the weighted cue value memory elements to be added (see Figure 14). The actual addition facts are well learned and do not contribute to errors. The two strategies use different retrieval criteria to retrieve the weighted cue values. In the recall-strategy, the only criteria is the name of the attribute, for example RANGE. Each time a weighted cue value for range is created it will have the name attribute of RANGE. All of these will match the retrieval criteria. The one that will actually be returned to complete the retrieval will be the one with the highest activation (subject to noise). In the store-in-goal strategy, the memory element is part of the goal, so the memory element itself can be the criteria and be retrieved directly from memory. However, it can still fail to be retrieved if it is not sufficiently active.

The order of the addition is randomized in the recall strategy only. This is a deficiency of the model. It should randomize the order for both strategies. The Argus model relies on noise in the conflict resolution process to randomly select one of six productions. Each of the six initiates a different order of the addends in the target score equation. When the target score has been calculated the goal is switched to the calculate threat value sub-goal.

5.4.2.4. Calculate Threat Value

The calculate threat value sub-goal maps the target score to the 1 – 7 threat value. Each radio button is labeled with the threat value it represents. Below each radio button is the range of target score values that are mapped to the radio button. The Argus model finds the location of the nearest radio button from the current focus of visual attention (this will vary depending on the

order in which the attributes were read). The visual location buffer will contain this location. The target score is used as a criteria to retrieve a ts->threat memory element which maps a threat value to a button location. If the retrieval is successful the Argus model will use that location to move visual attention and the cursor to the radio button. If the retrieval is not successful, then the Argus model will move visual attention to the location in the visual location buffer. The Argus model compares the calculated target score with the text of the radio button. If the target score is within the range indicated by the text, the Argus model will move the cursor to the location. If not it will move attention to the next radio button and repeat the comparison until the correct radio button is located. The Argus model issues a motor command to click the mouse. The Argus model also creates a ts->threat memory element to be retrieved in subsequent classifications. The Argus model thus attempts to learn a score to location mapping of the radio button. Ehret showed that object locations can be learned (Ehret 2000). The Argus model issues a command to the vision module to find the location of the enter button and commands the motor module to move the mouse to that location and click the enter button. An improvement to the Argus model would be to learn the location of the enter button similar to how it learns the location of the radio buttons and feedback screen (described in next section).

The target that was classified is retrieved and updated with the fact that it has been classified and the segment in which it was classified. When classification has been completed, feedback processing or target selection can be initiated. Conflict resolution will choose the unit task with the highest utility. The utility is a function of the experimental condition and the classification feedback subject parameter. In the no feedback conditions, the production that initiates target selection will always have a higher utility than the production that initiates feedback processing.

5.4.3. Feedback Processing.

The flow of the Feedback Processing unit task is shown in Figure 15.

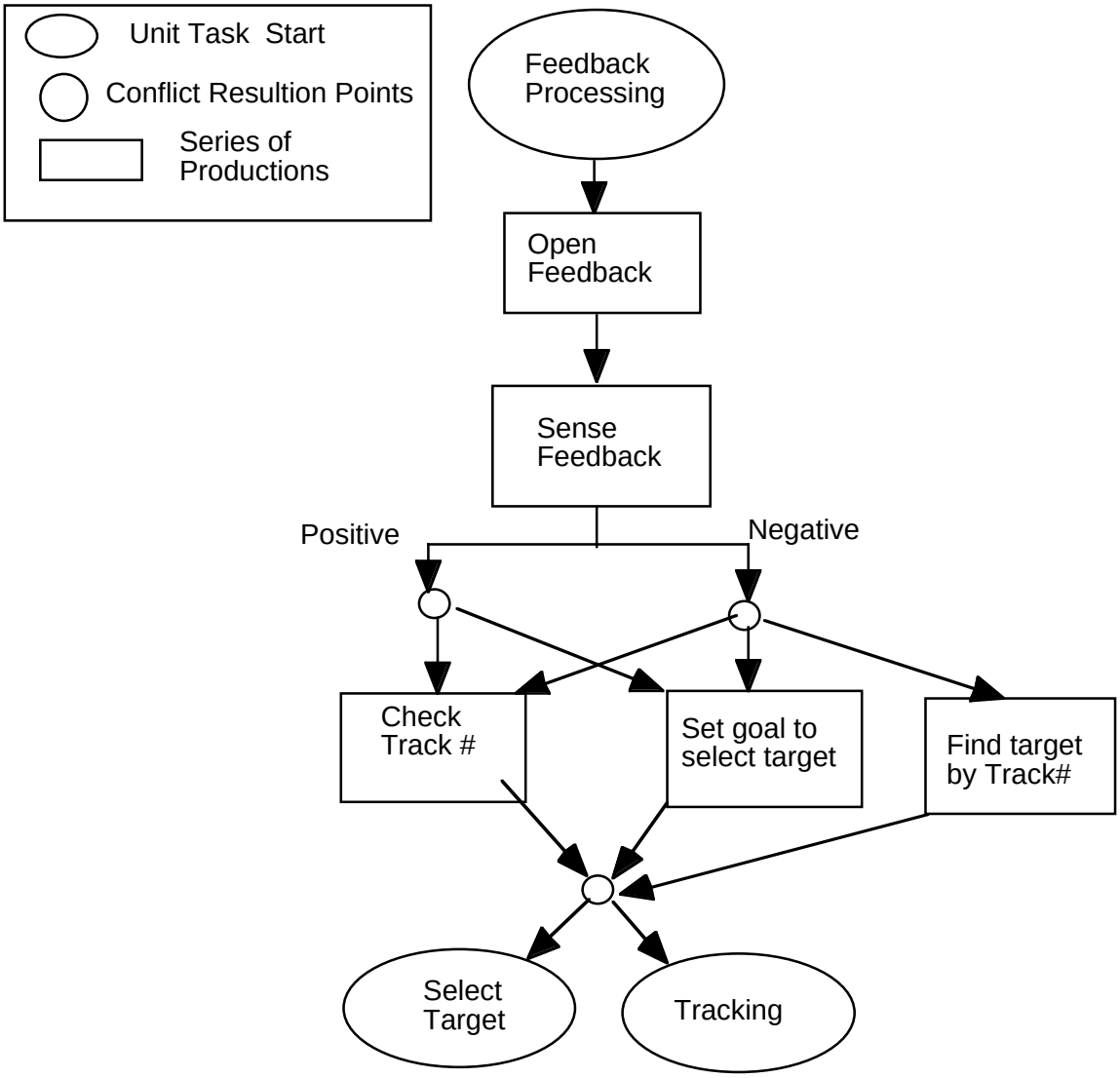


Figure 15. Flow of Feedback Processing unit task.

The feedback processing issues a command to the vision module to find the location of the feedback window and in parallel it tries to retrieve the location of the gray feedback window. The first operation to successfully complete provides the location for a move attention and move cursor operation. When the move cursor command completes, a mouse click command is issued to the motor processor. When the mouse click is complete the feedback window is uncovered. The Argus model issues a command to the vision module to locate the feedback icon. Two productions check the color feature at the location returned in the visual location buffer (see Figure 15). A blue color feature indicates a correct classification and a red color feature indicates an error. If the classification is correct the select target goal is retrieved and set to be the active goal. If the classification was incorrect, the Argus model does one of three things (see Figure 15). The choice is controlled by the reclassify and feedback track number subject parameters. The first possible action is to simply ignore the error and begin selecting the next target to be classified by retrieving the select-target unit task goal and putting it in the goal buffer. The second possible action is to look at the track number to strengthen the trace of which target is being processed and try to reclassify it. Looking at the track number has a perceptual motor cost since the track number is covered and requires an eye movement, mouse movement and a mouse click to uncover plus a movement of attention to read the number. Rather than incur this cost, the third possible action is not to look at the track number and depend on a memory retrieval to identify the track number of the target to be reclassified. However, a retrieval error may lead to reclassifying the wrong target.

5.4.4. Tracking Task

The flow of the tracking unit task in shown in Figure 16. The tracking task is initiated when either the tracking cursor is yellow or red or when the difference between the classification task score and the tracking score is greater than the threshold set for the current SHU. The initial

step of the tracking task is to check the pointing device by issuing a find location command to the vision module for the pointing device. If the pointing device is blue, two productions will match and compete for execution (see figure 16). One ends the tracking task and proceeds to the next unit task. The second initializes the tracking task. The utility of the productions is controlled by the tracking subject parameter. If this parameter is set to increase tracking, the utility of the second production will be higher. If the pointing device is yellow, two productions will match (see Figure 16). One will return to the Classify Target unit task and one will initiate tracking. They compete equally. If the pointing device is red, tracking will always be initiated.

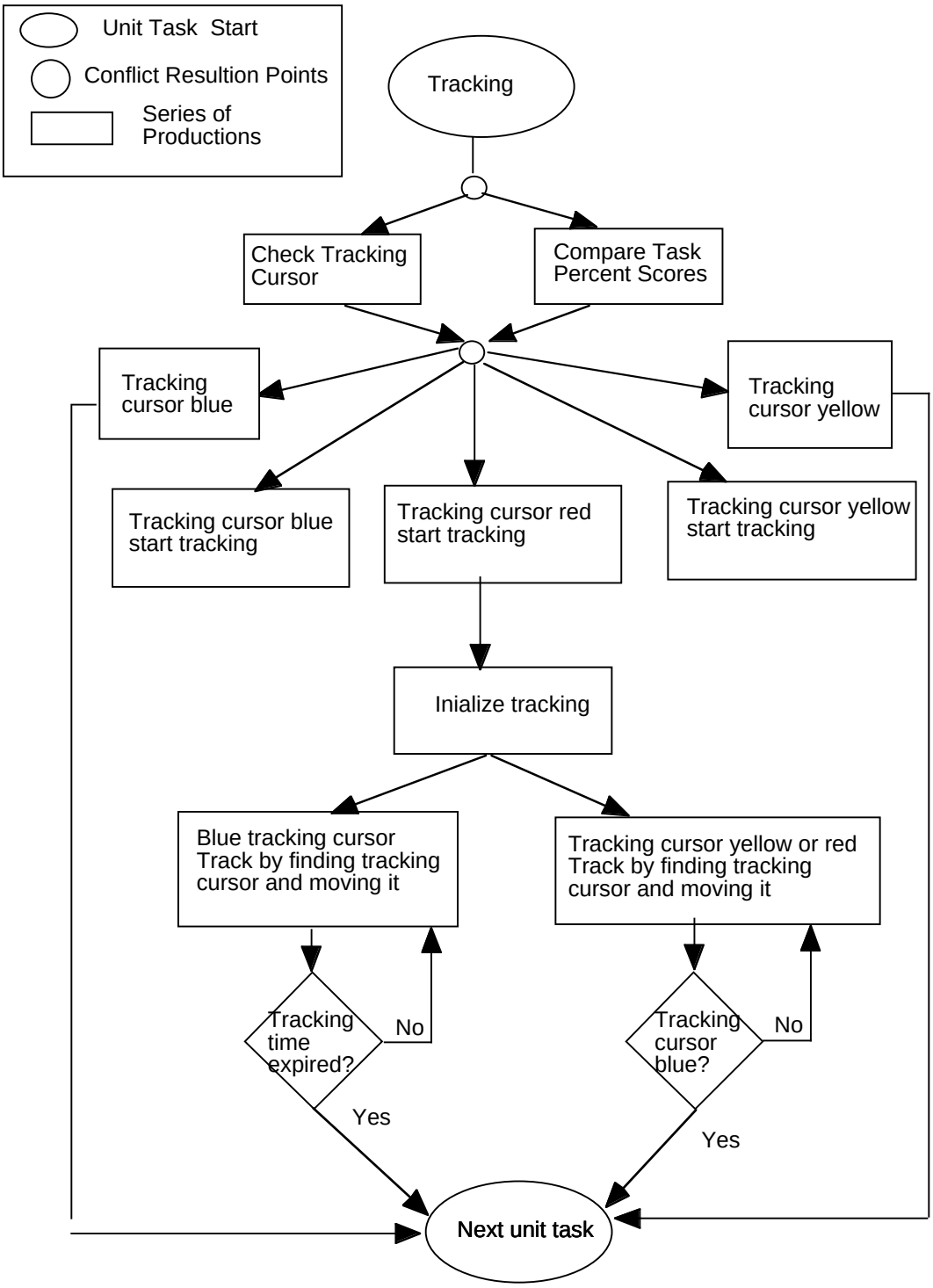


Figure 16 Flow of tracking unit task.

When the tracking task is initiated, a command is issued to the motor module to press the shift key, which causes the cursor to become the pointing device so that it will be controlled by the mouse. A find location command is issued to the vision module to determine the location of the plane. When this operation completes, visual attention is moved to the plane. A command is issued to the vision module to begin tracking the plane. The vision module will continuously update the location of the plane in its iconic memory until attention is shifted to another object. A command is also issued to the motor module to move the cursor to the object contained in the visual buffer. The Argus model then monitors the color of the pointing device while it issues commands to the motor module to move the cursor to the current plane location. The plane's location is automatically updated by the vision module. The Argus model continues tracking until a time limit imposed by the tracking subject parameter expires.

5.2. Model Execution

The ACT-R 5.0 architecture captures the parallelism between cognition, perception and motor processing found in human behavior by allowing concurrent buffer changes. The Argus model takes advantage of parallelism. Thirty-three of the productions perform two or more buffer operations. This sets a lower bound on the degree of parallelism achieved in the model, since parallel retrieval, shifts of attention and motor operations can be achieved in sequential productions if one operation is initiated before a previous one completes.

Each run of the Argus model completes one twelve minute scenario. Figure 17 shows the activity of the Argus model for one scenario in each experiment. The darkest area depicts the percent of the total production firings that initiated two or more buffer changes. Again this is a lower bound on the degree of parallelism achieved.

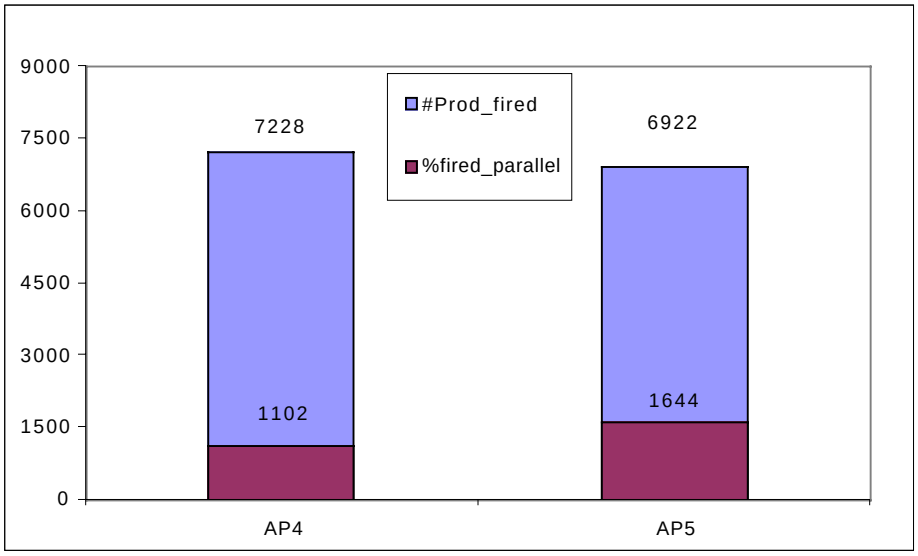


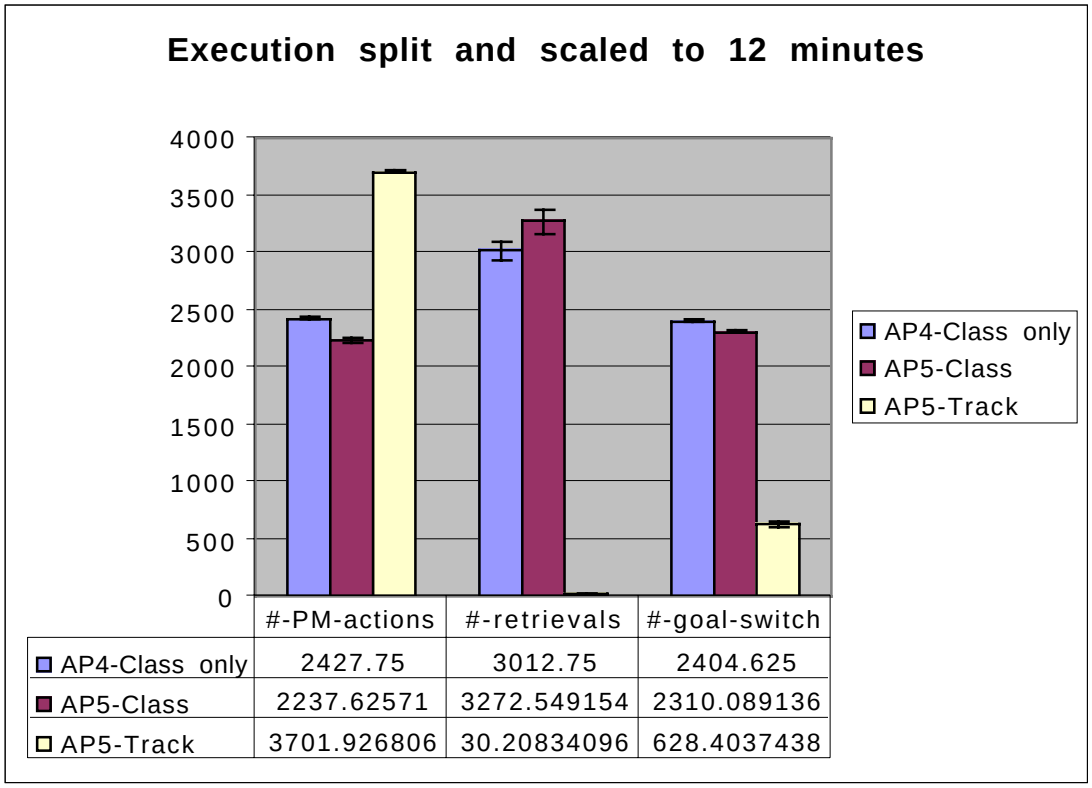
Figure 17. Mean number productions fired and mean number firing parallel actions.

Figure 17 shows that productions fire on about half the cycles, since in 12 minutes, 14,400 productions firings are possible. Idle time occurs when a retrieval, perceptual, or motor action is in progress and no other production is eligible for firing.

The mean number of retrievals, perceptual-motor actions and goal switches that are executed by the Argus model during a single scenario are shown in Figure 18. The data for AP4 pertains to the classification task only and the data is taken from 12 minutes of model execution. The data for AP5 is divided into two series, one for the classification task and one for the tracking data. The means for each series are scaled to 12 minutes in order to be comparable to the AP4 data. The scaling was done by multiplying the mean by the scaling factor. The scaling factor is 12 divided by the number of minutes spent in each task. In the dual task environment (AP5), the Argus model performs about the same on the classification task as in the single task environment.

It performs more perceptual motor operations than in the single task environment (AP4). The figure shows that the tracking task execution consists of considerably more perceptual-motor actions than cognitive actions. This is not surprising since the addition of the tracking task was intended to increase perceptual motor workload.

The figure also provides some evidence that the addition of the perceptual-motor task did not change the strategies used in the Classification task, since, per unit of time, the number of perceptual-motor actions, the number of retrievals, and the number of goals switches were about equal in the two experiments.



	#-PM-actions	#-retrievals	#-goal-switch
AP4-Class only	2427.75	3012.75	2404.625
AP5-Class	2237.62571	3272.549154	2310.089136
AP5-Track	3701.926806	30.20834096	628.4037438

Figure 18. Mean number of perceptual-motor actions, retrievals and goal buffer changes.

The final measure of execution activity, shown in Figure 19, is number of episodic memory traces created. In the single task environment (AP4) a chunk is created about every 450 ms, whereas in AP5, a chunk is created about every 370 ms. The increase in the number of chunks in the dual task environment is due to the creation of location traces. This again shows that adding a perceptual-motor task has an impact on cognitive operations that can be quantified by the model.

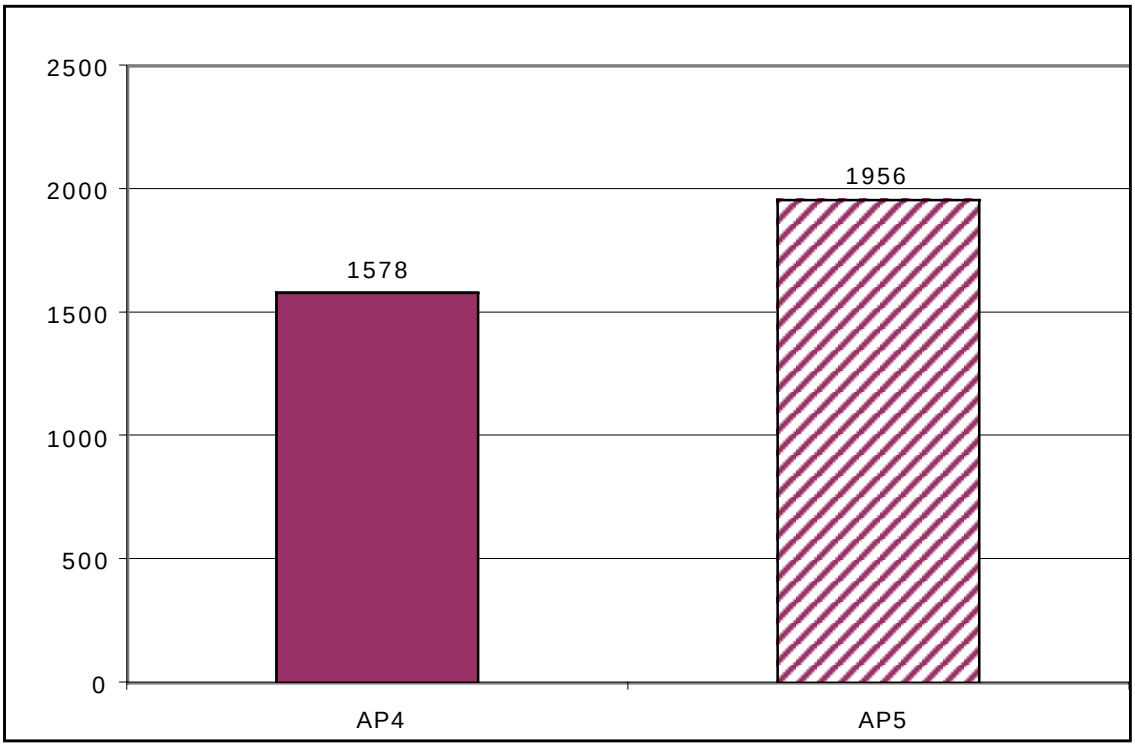


Figure 19 Mean number of episodic memory traces created.

6. Results

The claim that the Argus model is successful in simulating human performance on the primary classification task is validated by comparing the Argus model data to the human data in the single task experiment (AP4). The ability of the Argus model to predict human performance is shown by comparing model data to human data on the classification task when the secondary tracking task is added in the dual task experiment (AP5). The results show that the degradation of human performance can be predicted on the basis of the change in the Argus model performance. The results are presented for a variety of output measures. The output measures are intended to represent a decomposition of the classification task into unit task level components. The Argus model data comes from running the Argus model 96 times (24 model-subjects * 4 scenarios) in each experiment. The human data comes from 4 scenarios for each of the 24 subjects in each experiment. A model-subject did the same four scenarios as a human subject. Performance graphs and a statistical analysis are given for each output measure.

6.1 Statistical Analysis

The statistical method used to compare model and human data is a repeated measures Analysis of Variance (ANOVA). The between-subjects condition is human vs. model. The within-subjects condition is a 2 by 2 design. The two factors represent the interface conditions manipulated in the two experiments. The first factor, classification, has two levels, dot vs. no-dot. The second factor, feedback, also has two levels, fb vs. no-fb. A good fit of the Argus model to the human data is indicated by no main effect of human vs. model.

Unlike research endeavors where the intent is to show an effect, the goal of a computational cognitive model is a veridical simulation of human performance. Thus the burden is to demonstrate no differences in performance of the Argus model and the human subjects. In the hypothesis testing framework, a significant result implies that the null hypothesis can be rejected in favor of the research hypothesis, but here the research hypothesis is the null hypothesis. Thus, if the result of a test on an output measure is not significant, all that can be said is that the null hypothesis of no difference can not be rejected.

The results reported in this chapter include a post-hoc power analysis when the ANOVA is inconclusive (i.e. non-significant p-value) in order to provide additional evidence that the SHU data matches the human data. The power analysis will show that any differences between human and model subject on the output measures are indeed small enough not to be considered important. The power analysis includes the post-hoc power of the ANOVA F test, the effect size, and the sample size that would have been required to detect any difference between the human and model subjects. These three are interrelated. Power is the probability of correctly rejecting a false null hypotheses. When there is significant differences between humans and the model, power should be high enough to detect these differences. Put another way, if there are big differences between the human and model data then power should be high. Conversely, if the differences between the groups are small then power will be low. Power indicates a high degree of overlap between the sampling distribution of the means under the two conditions. Low power indicates that the distributions lie on top of each other. The separation of the distributions is related to effect size. Effect size is defined as the difference in means divided by the standard deviation, thus power and effect size are related and indicate the degree of difference between the groups. Effect size is a distance measure on the means independent of sample size. A small effect size calculated in a post-hoc power analysis will provide evidence that any differences between

the Argus model and human are small differences. For an F-test (ANOVA) an effect size of 0.1 is considered small, 0.25 medium and 0.4 large (Cohen 1988). Given the effect size, the number of samples (N) it would have taken to detect a significant difference can be calculated. When the ANOVA does not indicate a significant result, then the post-hoc power, effect size, and N will be reported to support the claim that the model is a veridical simulation.

Clustered column graphs are presented for each output measure. The ordinate for each output measure represents the mean of the measure for the indicated category. The abscissa represents the four experiment conditions (i.e. previously classified indicator (the dot) on or off vs. immediate classification feedback (right or wrong) shown or not shown). The mean values are displayed for human and model data in the single task experiment (AP4) and in the dual task experiment (AP5). A means table is also provided with each graph. The error bars represent standard error for each category.

6.2. Outcome Measures

Overall task performance is measured by the total score the subject received. The total score is the ratio of the number of targets correctly classified to the total number of possible classifications expressed as a percentage. As discussed above, in order to strengthen the claim that the SHUs are accurately simulating the human users at a level that is of value in Human Computer Interaction, they must not only match performance at the task level but must match at the unit task level as well. To validate that the SHUs meets this goal a number of finer grained measures have been analyzed and are presented in this chapter. Table 9 summarizes these measures and Figure 20 depicts the hierarchical relationship of the measures.

Table 9. Unit Task Output Measures.

Name	Unit Task	Type	Description
Total selects	Select Target	Quantity	Number of targets that were clicked on (selected)
Initial selects	Select Target	Quantity	Number of targets that were not classified when selected
# Selects - already classified	Select Target	Quantity	Number of targets that were classified when selected
Time target to target	Select Target	Duration	Time from clicking on one target, deciding not to classify it and to clicking on the next
Total classifications	Classify Target	Quantity	Number of targets for which a classification was made
Total correct	Classify Target	Quantity	Number of correct classifications
Initial incorrect	Classify Target	Quantity	Number of classifications that were incorrect and the target had not been previously classified
Total reclassifications	Classify Target	Quantity	Number of times a target was classified and it had previously been classified
Classification time	Classify Target	Duration	Time from selecting target to pressing radio button with no intervening actions
Number of feedback accesses	Feedback Processing	Quantity	Number of times feedback window accessed
Feedback time	Feedback Processing	Duration	Total time spent in feedback processing. Time from opening feedback window to leaving window
Number of track number accesses	Feedback Processing	Quantity	Number of times the track number displayed in the feedback window was accessed

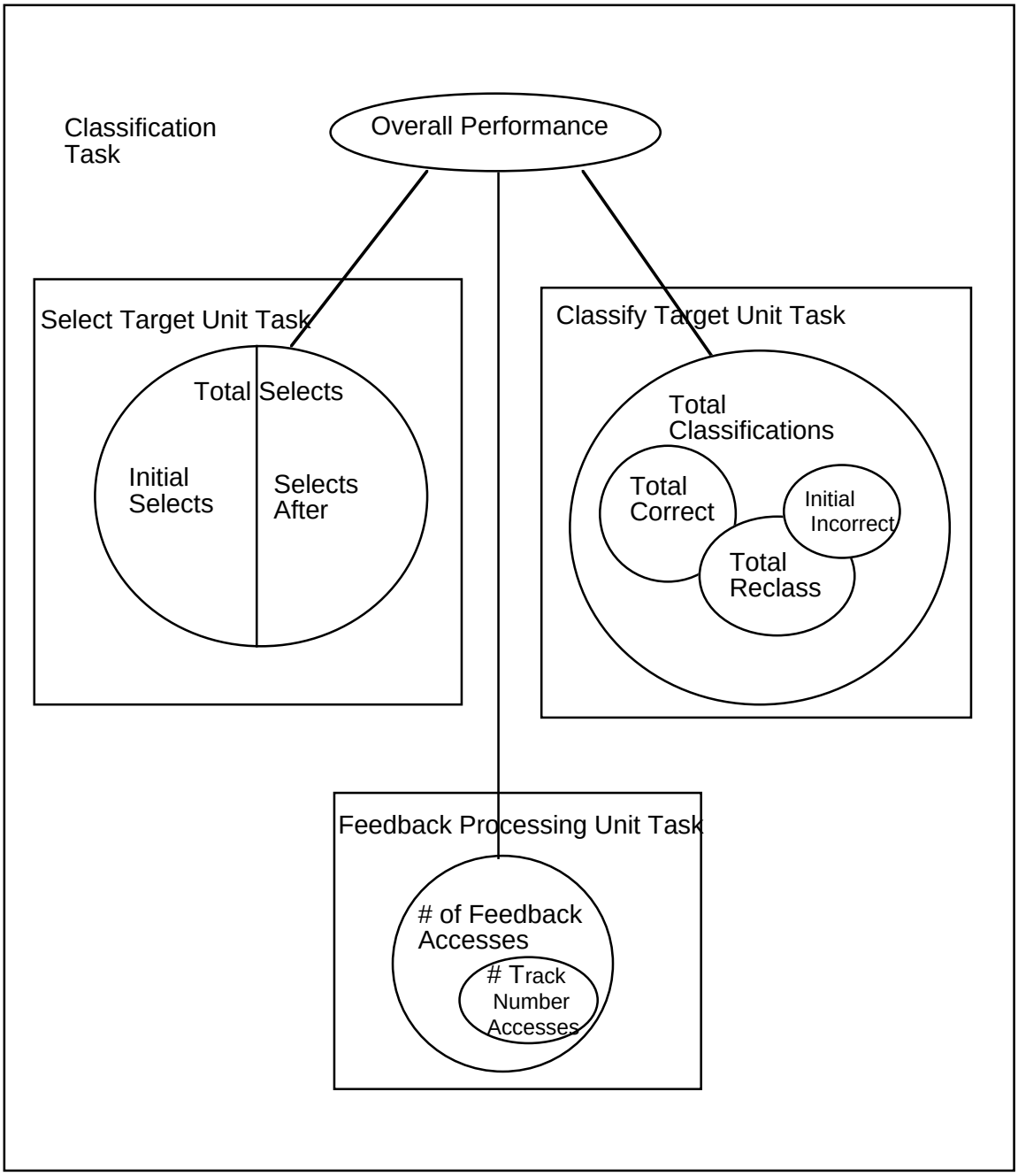


Figure 20. Hierarchical relationship of output measures for Classification task.

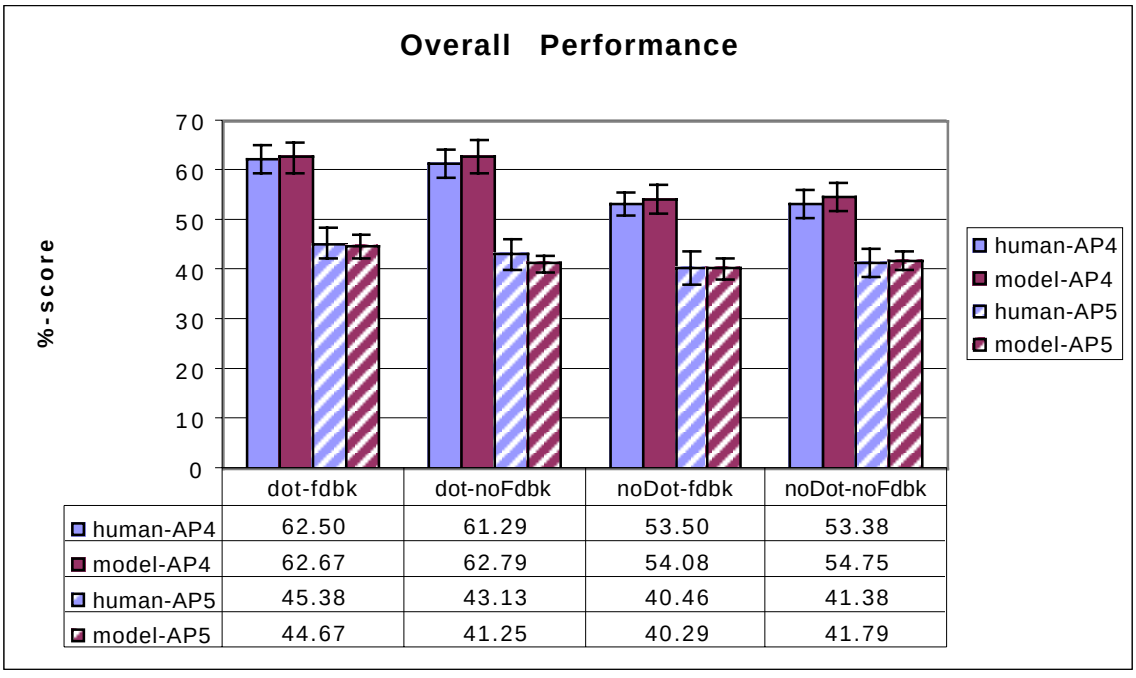
6.3. Predicting Performance on Classification Task

6.3.1. Overall Performance

The overall performance on the classification task is calculated by:

%-score = (number-correct / total-opportunities) * 100

Number-correct is the total number of times that a target classification was made and the classification was correct. Total-opportunities is the total number of classification opportunities where an opportunity is a target appearing in a sector (i.e. each target must be classified once for each sector in which it appears). Figure 21 compares the aggregate overall performance of human and model in the single task condition and in the dual task condition.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	62.50	61.29	53.50	53.38
model-AP4	62.67	62.79	54.08	54.75
human-AP5	45.38	43.13	40.46	41.38
model-AP5	44.67	41.25	40.29	41.79

Figure 21. Mean %-score for human and model for AP4 and AP5.

In the single-task experiment (AP4), the difference between model and human data was not significant, $p = 0.81$ ($F(1,46) = 0.06$, $MSE = 665.31$). The post-hoc power analysis resulted in an effect size of 0.032 and power of 0.056. To detect any differences between the SHUs and human subjects with a power of 0.8, a total sample size of 7,668 would have been required. The ANOVA also showed that there were no significant interactions involving the subject type. As predicted, the main effect of dot-noDot was significant $p < 0.0001$ ($F(1, 46) = 50.13$, $MSE = 67.32$). The main effect of feedback was not significant.

The ability to predict human performance when the secondary task was added in AP5 is also shown in Figure 21. The main effect of model vs. human was not significant $p = 0.86$ ($F(1,46) = 0.03$, $MSE = 544.28$). No interactions involving subject type were significant. The power analysis resulted in a power of 0.053 and a small effect size of 0.023. 14,712 subjects would have been required to obtain a significant difference.

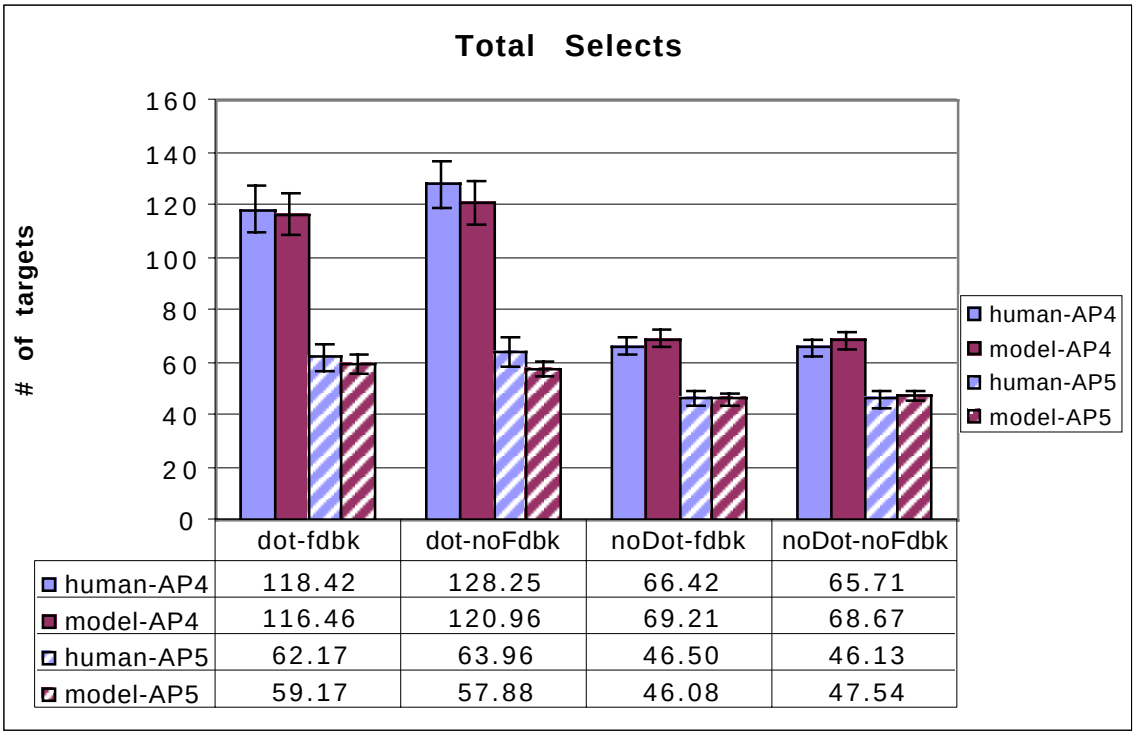
Across all conditions, performance on the primary task was degraded for the SHUs and the human subjects. Like AP4, in AP5 the main effect of the dot-noDot was significant $p = 0.0004$ ($F(1, 46) = 9.12$, $MSE = 36.12$), and the main effect of immediate feedback was not significant. There were no significant interactions involving the subject type.

6.3.2. Select Target Unit Task Results

This section discusses the results for the Select Target unit task. The main quantity result is the total number of target selections. This measure is factored into the number of target selections when the target had not been previously classified and the number of target selections when the target was previously classified. The time to select a target with no intervening actions is the time measure.

6.3.2.1. Total Selections.

Total selects represents the total number of targets selected during the course of a scenario. This number includes targets that were unclassified at time of selection and targets that had already been classified at time of selection. Figure 22 shows a good fit of the Argus model data to the human data in AP4. The main effect of model vs. human was not significant, $p = 0.91$ ($F(1,46) = 0.013$, $MSE = 2723.40$). No interactions involving subject type were significant. The power analysis resulted in a power of 0.051, a small effect size of 0.015. 33,098 subjects would have been required to obtain a significant result.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	118.42	128.25	66.42	65.71
model-AP4	116.46	120.96	69.21	68.67
human-AP5	62.17	63.96	46.50	46.13
model-AP5	59.17	57.88	46.08	47.54

Figure 22. Mean total number of target selections per scenario.

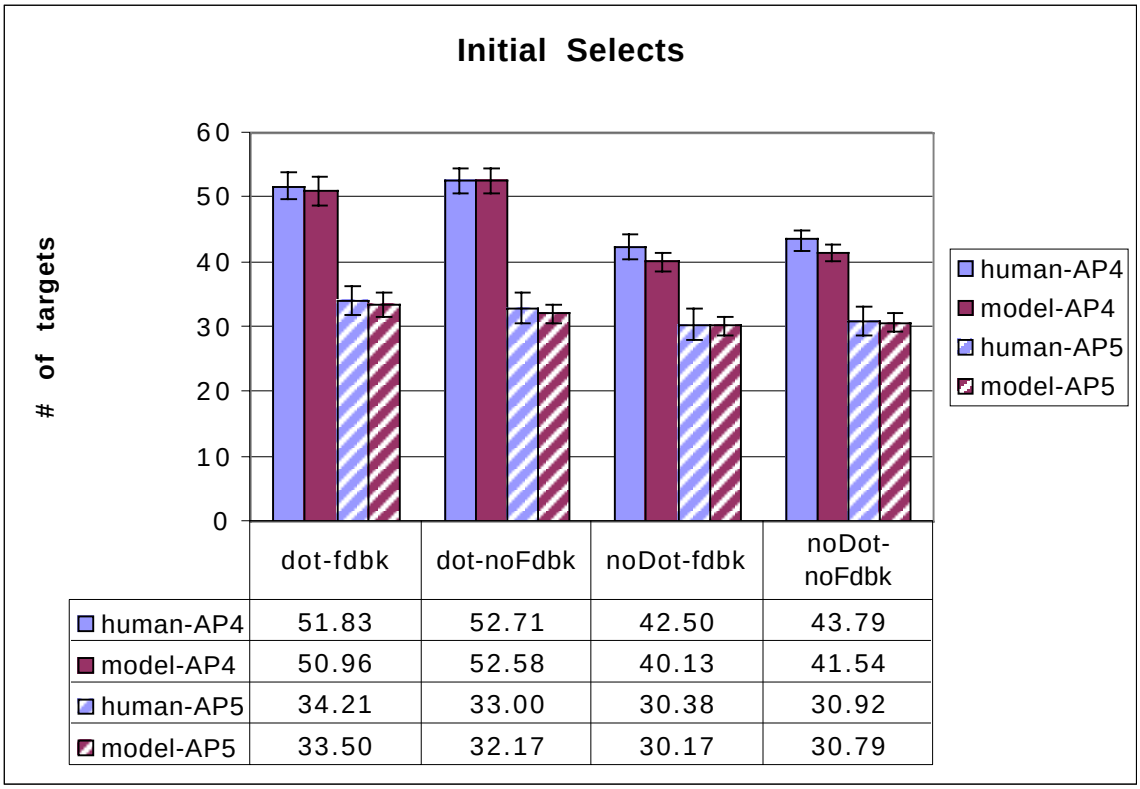
In AP4, the Argus model was able to capture the main effect of dot-noDot, $p < 0.0001$ ($F(1,46) = 180.79$, $MSE = 760.54$) and subject type did not interact with dot-noDot. The main effect of feedback was not significant.

In AP5, the total number of targets selected drops with the addition of the tracking task. SHUs and human subjects select fewer targets in all conditions. The ANOVA, $p = 0.66$ ($F(1,46) = 0.19$, $MSE = 1027.06$), indicates no significant difference between SHUs and human subjects. The power analysis calculated a power of 0.071, a small effect size of 0.060 and 2,234 subjects would have been required to produce a significant result. No interactions involving subject type were significant.

As in AP4, the Argus model captured the main effect of dot-noDot which was significant, $p < 0.0001$ ($F(1,46) = 68.86$, $MSE = 141.14$) and dot-noDot and subject type did not interact. The main effect of feedback was not significant.

6.3.2.2. Initial Selects

The total number of target selections can be decomposed into the number of targets that were not classified at the time of selection and the number there were already classified. Initial selects represents the former. Since the Classification task is a time pressured task, the higher the number of selections of targets that need to be classified should lead to better performance. As shown in Figure 23, the fit of SHU data and human data in AP4 is quite good. The main effect of subject type is not significant, $p = 0.55$ ($F(1,46) = 0.37$, $MSE = 255.37$). No interactions involving subject type were significant. The power analysis resulted in a power of 0.09, a small effect size of 0.082, and 1,182 subjects would have been required to detect a significant difference.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	51.83	52.71	42.50	43.79
model-AP4	50.96	52.58	40.13	41.54
human-AP5	34.21	33.00	30.38	30.92
model-AP5	33.50	32.17	30.17	30.79

Figure 23. Mean number of targets selected that were not classified when selected.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 201.02$, MSE = 24.03) and since the Argus model did not interact with dot-noDot the Argus model was able to capture the interface manipulation. The main effect of feedback was not significant.

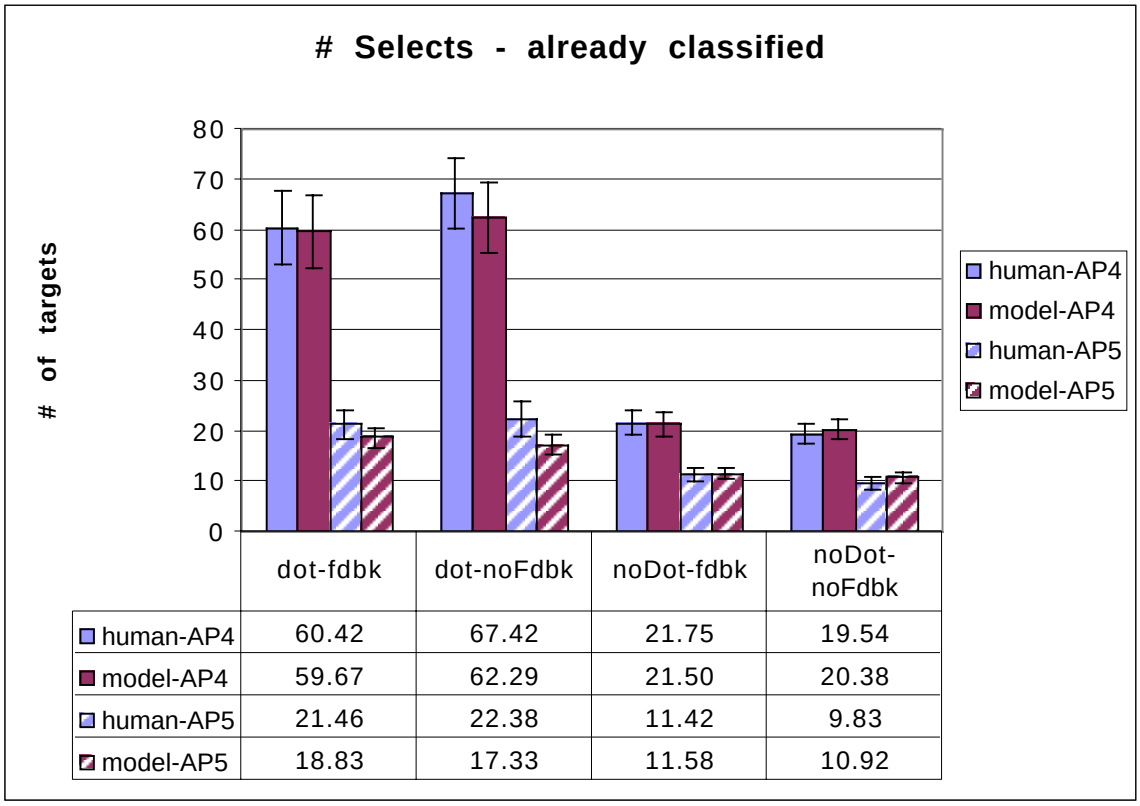
In AP5, the main effect of subject type was not significant, $p = 0.85$ ($F(1,46) = 0.034$, MSE = 309.92). The power analysis resulted in a power of 0.054 with an effect size of 0.025. 12,764 subjects would have been required to detect a significant difference.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 18.52$, MSE = 18.29) and since the Argus model did not interact with dot-noDot the Argus model was able to capture the interface manipulation. The main effect of feedback was not significant.

6.3.2.3. Selects – already classified

Selecting targets that not do require classification is predicted to have a negative effect on performance, especially in the noDot condition. In fact the purpose of the dot vs. noDot manipulation was to study the effect of knowledge in the world vs. knowledge in the head. In the dot condition, access to the classification status of a target is provided via an eye movement and two motor operations, whereas, in the noDot condition memory retrievals are required to access the status.

In AP4, the main effect of subject type was not significant, $p = 0.81$ ($F(1,46) = 0.056$, MSE = 1488.79). This represents a good fit of the Argus model data to the human data. The variability is higher in the dot condition than in the noDot condition indicating that the subject may be using a mix of strategies. Figure 24 shows that the Argus model captures this trend in variability. There were no significant interactions involving subject type. The power analysis calculated a power of 0.056, a small effect size of 0.031, and 8,276 subjects would be required to achieve a significant result at 0.8 level.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	60.42	67.42	21.75	19.54
model-AP4	59.67	62.29	21.50	20.38
human-AP5	21.46	22.38	11.42	9.83
model-AP5	18.83	17.33	11.58	10.92

Figure 24. Mean number of target selections when target was classified.

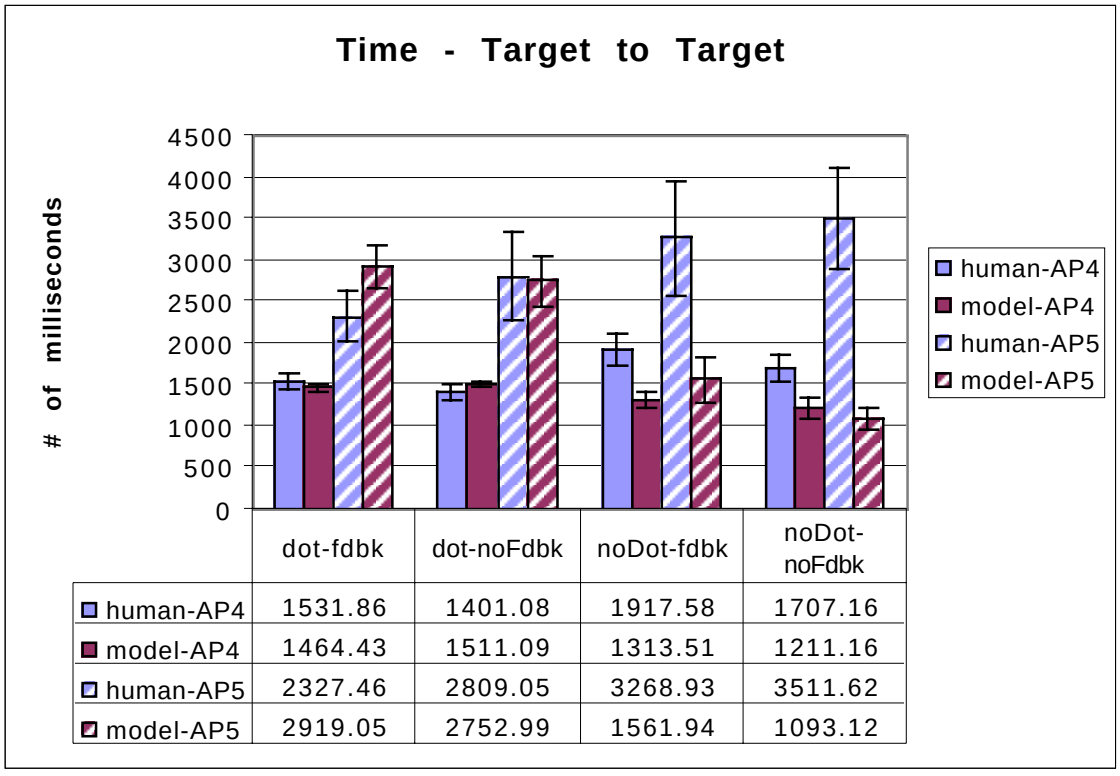
Like the previous measures, the main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 143.46$, MSE = 580.60). The Argus model was able to capture the interface manipulation. The main effect of immediate feedback was not significant.

In AP5, the main effect of subject type was not significant, $p = 0.48$ ($F(1,46) = 0.50$, MSE = 248.95). No interactions involving subject type were significant. The power analysis resulted in a power of 0.10, a small effect size of 0.089, and 1,012 subjects would be required to obtain a significant result.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 51.99$, MSE = 75.82). The main effect of immediate feedback was not significant.

6.3.2.4. Target Search – Target to Target

This measure is the number of milliseconds between successive target selections (measured from mouse click to mouse click) with no intervening actions such as making a classification or getting help. As such it is a pure measure of searching for the next target to process. As seen in Figure 25, the Argus model is faster than the humans in the no-dot condition which results in a poor fit of the model.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	1531.86	1401.08	1917.58	1707.16
model-AP4	1464.43	1511.09	1313.51	1211.16
human-AP5	2327.46	2809.05	3268.93	3511.62
model-AP5	2919.05	2752.99	1561.94	1093.12

Figure 25. Mean time in milliseconds from selecting a target then selecting another target with no intervening actions over the course of a scenario.

In AP4, the main effect of subject type is significant as can be seen in Figure 21 and this is verified by the ANOVA, $p = 0.024$ ($F(1,46) = 5.437$, $MSE = 603907.38$). Subject type does

interact with dot-noDot, $p < 0.0001$ ($F(1,46) = 19.01$, MSE = 201702.17). Neither the main effect of dot-noDot or immediate feedback are significant.

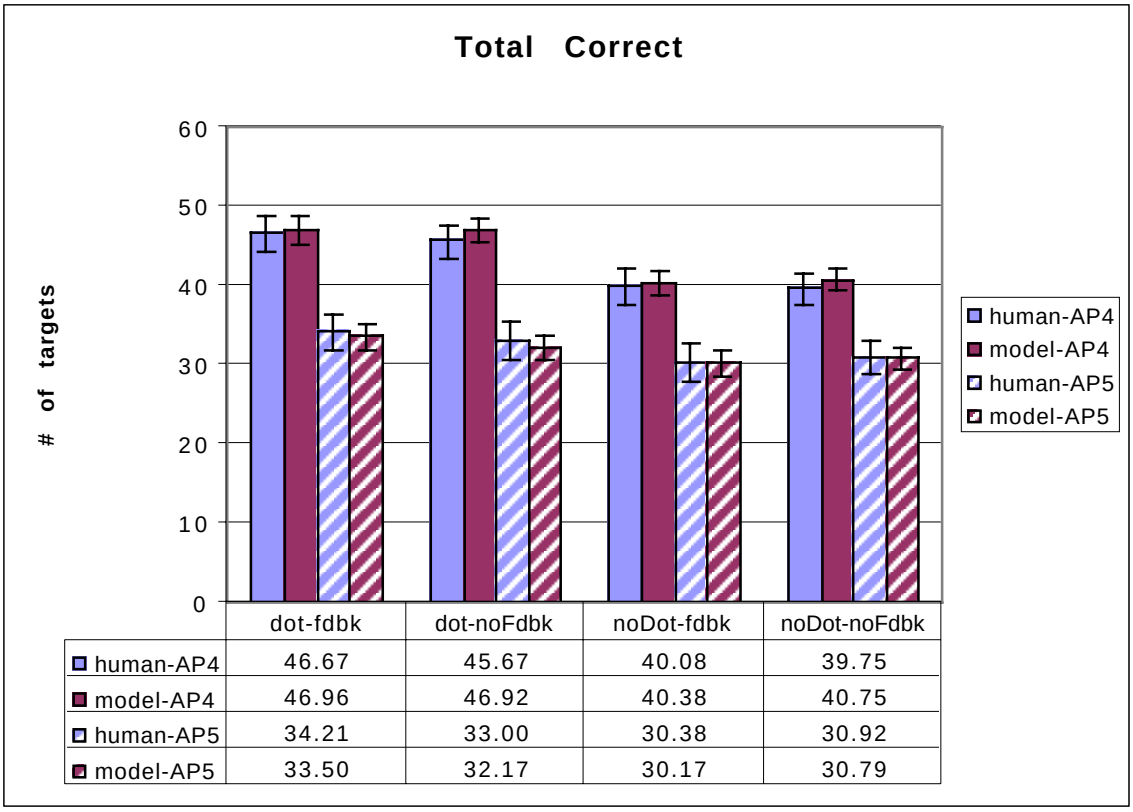
In AP5, the main effect of subject type was significant. The ANOVA resulted in $p = 0.025$ ($F(1,46) = 5.389$, MSE = 7173921.86). Subject type did not interact with dot-noDot or immediate feedback. Like AP4, neither the main effect of dot-noDot or immediate feedback are significant.

6.3.3 Classify Target Unit Task Results

This section discusses the results for the Classify Target unit task. The main quantity result is the total number of correct classifications. The other quantity measures are the number of classification initially incorrect, the total number of classifications and total number of reclassifications. The time measure is the time to classify a target.

6.3.3.1. Total Correct

The number of total correct is the total number of classifications made that were correct regardless of the classification status of the target. It reflects both the quantity of targets processed and the accuracy of executing the classification algorithm. In AP4, as Figure 26 shows, the main effect of subject type is not significant, $p = 0.80$ ($F(1.46) = 0.067$, MSE = 356.93). No interactions involving subject type were significant. The power analysis resulted in a power of 0.057, a small effect size of 0.035, and 6,522 subjects would have been required to produce a significant result.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	46.67	45.67	40.08	39.75
model-AP4	46.96	46.92	40.38	40.75
human-AP5	34.21	33.00	30.38	30.92
model-AP5	33.50	32.17	30.17	30.79

Figure 26. Mean number of total correct target classification made during a scenario.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 70.61$, MSE = 27.09). The main effect of immediate feedback was not significant.

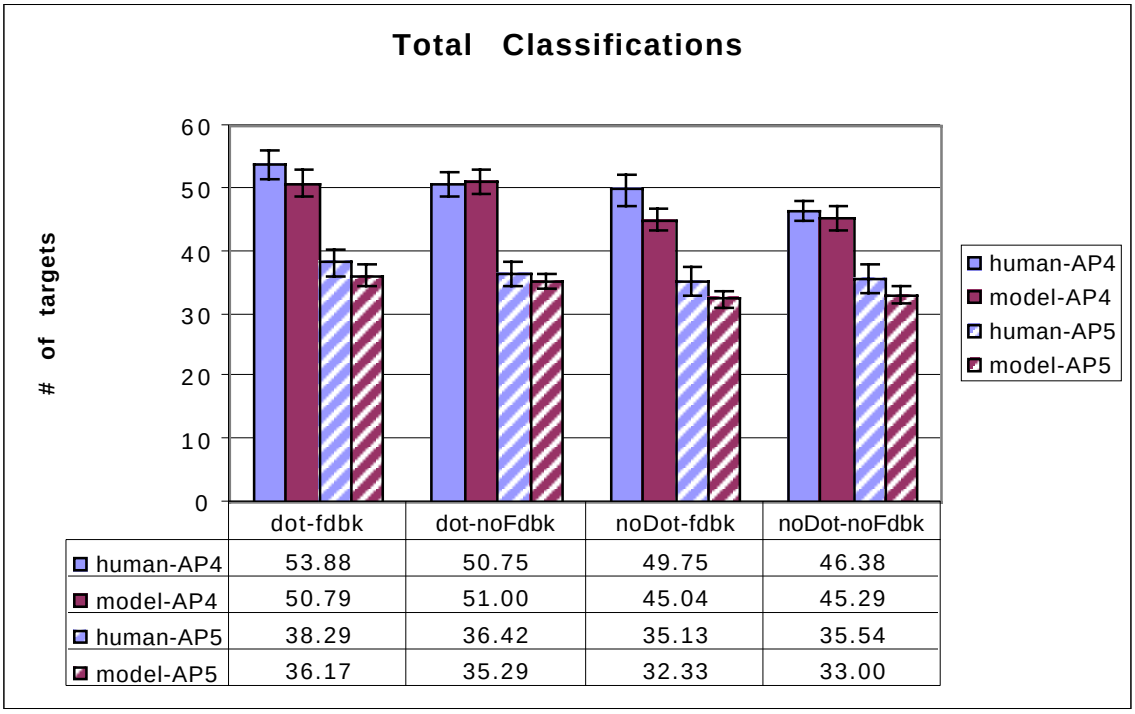
In AP5, the Argus model again had a good fit to the human data. The main effect of subject type was not significant, $p = 0.85$ ($F(1,46) = 0.034$, MSE = 309.92). No interactions involving subject type were significant. The power analysis resulted in a power of 0.054, a small effect size of 0.025, and 12,764 subjects would have been required to produce a significant result.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 18.52$, MSE =18.29). The main effect of immediate feedback was not significant.

6.3.3.2. Total Classifications

The total number of classifications includes first time target classifications and reclassifications. It contributes to total performance in that time spent on classifying already classified targets is wasted time. Figure 27 shows a good fit of the Argus model data to human data in AP4. The main effect of subject type is not significant, $p = 0.41$ ($F(1,46) = 0.70$, MSE = 320.69). No interactions involving subject type were significant. The power analysis resulted in a power of 0.12 with a small effect size of 0.11 and 652 subjects would have been required to obtain a significant result.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 53.47$, MSE = 22.35). The main effect of immediate feedback was not significant.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	53.88	50.75	49.75	46.38
model-AP4	50.79	51.00	45.04	45.29
human-AP5	38.29	36.42	35.13	35.54
model-AP5	36.17	35.29	32.33	33.00

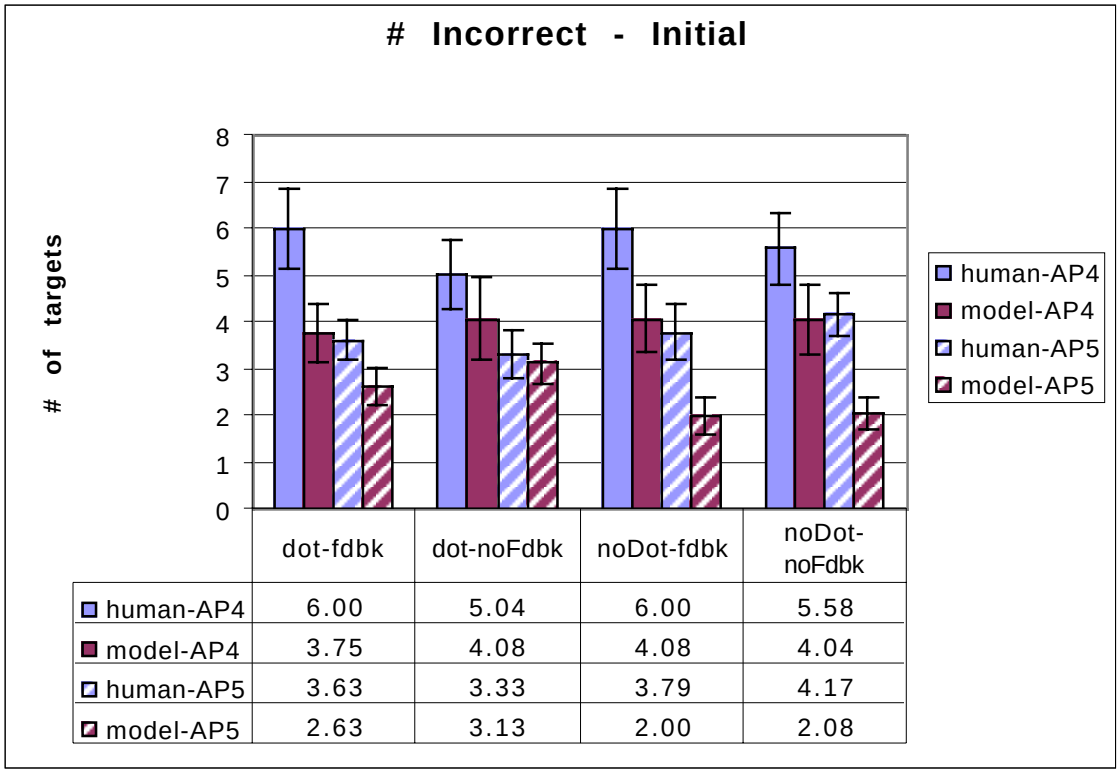
Figure 27. Mean number of total target classification made during a scenario.

In AP5, the main effect of subject type was not significant, p = 0.38 (F(1,46) = 0.80, MSE = 277.97). No interactions involving subject type were significant.

In AP5, the main effect of dot-noDot was significant, p = 0.0002 (F(1, 46) = 16.91, MSE = 18.34). The main effect of immediate feedback was not significant.

6.3.3.3. Initial Incorrect

Initial Incorrect is the number of targets that were classified incorrectly. It does not include incorrect reclassifications. It reflects the accuracy of executing the classification algorithm. In AP4, the repeated measures ANOVA produced a marginally significant effect for subject type, p = 0.076 (F(1.46) = 3.30, MSE = 40.35). No interactions involving subject type were significant. The power analysis resulted in a power of 0.41 with a small to medium effect size of 0.22 and 170 subjects would have been required to produce a significant result. Figure 28 reflects this marginal fit of the Argus model to the human data. The Argus model was more accurate than the human subjects.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	6.00	5.04	6.00	5.58
model-AP4	3.75	4.08	4.08	4.04
human-AP5	3.63	3.33	3.79	4.17
model-AP5	2.63	3.13	2.00	2.08

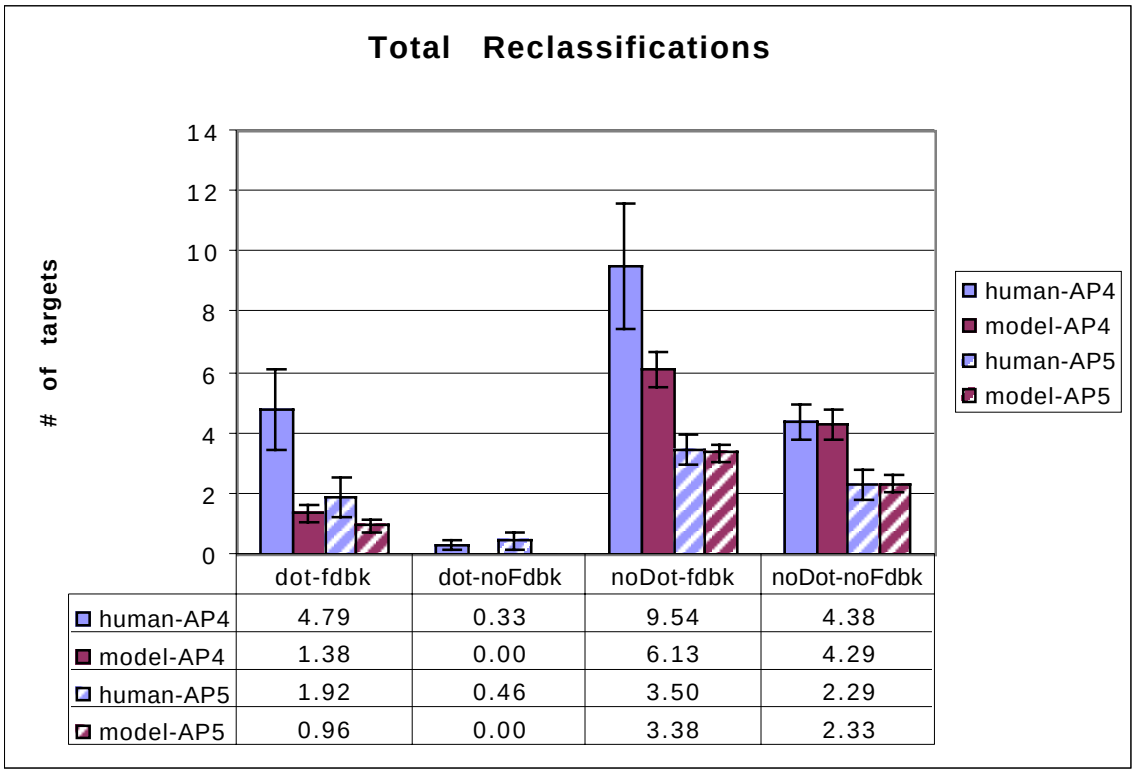
Figure 28. Mean number of initial classifications that were incorrect made during a scenario.

In AP5, the Argus model did not have a good fit to the human data. The main effect of subject type was significant, p = 0.0092 (F(1,46) =7.4, MSE =10.47). Subject type did interact with dot-noDot, p = 0.0068 (F(1,46) = 8.02, MSE = 2.66). The main effect of dot-noDot was not significant. The main effect of immediate feedback was not significant.

6.3.3.4. Total Reclassifications

The total reclassifications measure is the total number of times an already classified target was reclassified, regardless of whether the current classification was correct or incorrect. That is, the subject or model may be reclassifying the target because they had previously misclassified it and the immediate feedback indicated the error or they may be reclassifying a target in the noDot condition and do not remember that they had already classified the target.

In AP4, the ANOVA, p = 0.05 (F(1,46) = 4.06, MSE = 38.85) indicates a poor fit of the Argus model to the human data seen in Figure 29. The power analysis yields a power of 0.44, with a medium effect size of 0.27, and 114 subjects would be required to achieve a significant result at the 0.8 level. Subject type interacted marginally with feedback, p = 0.062 (F(1,46) = 3.67, MSE = 33.64).



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	4.79	0.33	9.54	4.38
model-AP4	1.38	0.00	6.13	4.29
human-AP5	1.92	0.46	3.50	2.29
model-AP5	0.96	0.00	3.38	2.33

Figure 29. Mean number of total reclassifications that were made during a scenario.

The main effect of the dot-noDot is significant, p < 0.0001 (F(1,46) = 128.81, MSE = 7.41). The main effect of immediate feedback is also significant, p = 0.0004 (F(1,46) = 14.69, MSE = 33.64).

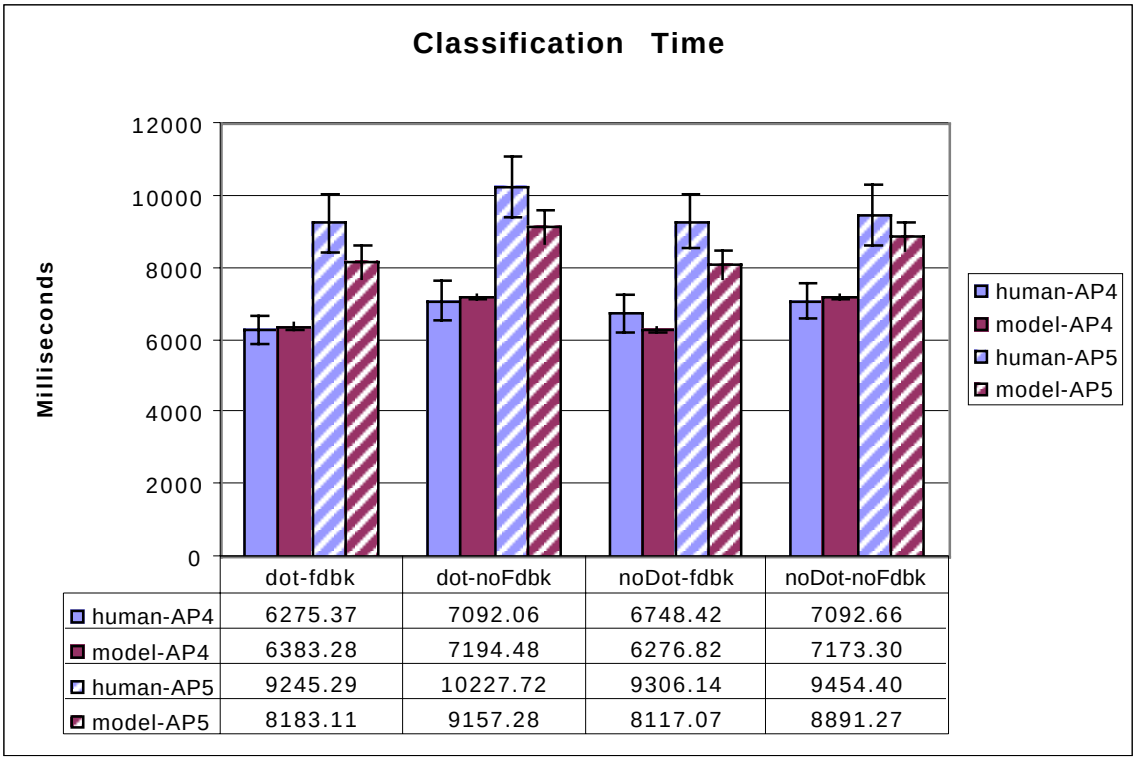
In AP5, the fit of the Argus model to the human data is good as reported by the ANOVA, $p = 0.33$ ($F(1,46) = 0.97$, $MSE = 6.97$). No interactions involving subject type were significant. The power analysis produced a power of 0.153, a small effect size of 0.11, and 630 subjects would have been required to obtain a significant result.

The main effect of dot-noDot was significant, $p < 0.0001$ ($F(1,46) = 66.65$, $MSE = 3.02$). The main effect of immediate feedback was also significant, $p < 0.0001$ ($F(1,46) = 32.73$, $MSE = 1.996$).

6.3.3.5. Classification Time

Classification time is the time from selecting a target to pressing the radio button indicating the threat value of the target with no intervening actions such as getting help. It is intended to represent the pure time to calculate the classification algorithm.

In AP4, Figure 30 shows that the fit of the Argus model data to the human data is good. The ANOVA reports no significant effect of subject type, $p = 0.924$ ($F(1,46) = 0.009$, $MSE = 8777041.5$). No interactions involving subject type were significant. The power analysis indicated a power of 0.05, a small effect size of 0.02, and 24,228 subjects would have been required to achieve this result.



	dot-fdbk	dot-noFdbk	noDot-fdbk	noDot-noFdbk
human-AP4	6275.37	7092.06	6748.42	7092.66
model-AP4	6383.28	7194.48	6276.82	7173.30
human-AP5	9245.29	10227.72	9306.14	9454.40
model-AP5	8183.11	9157.28	8117.07	8891.27

Figure 30. Mean time in milliseconds to classify a target. The time from selecting the target to pressing the radio button.

The main effect of dot-noDot was not significant. The main effect of feedback was significant, $p < 0.0001$ ($F(1,46) = 34.99$, MSE = 586514.80).

In AP5, the fit of the Argus model to the human data is good to moderate. The ANOVA reports no significant effect of subject type, $p = 0.25$ ($F(1,46) = 1.34$, MSE =25268181.5). No interactions involving subject type were significant. The power analysis indicated a power of 0.19, a small effect size of 0.18, and 236 subjects would have been required to achieve this result.

The main effect of dot-noDot was not significant. The main effect of feedback was significant, $p = 0.0063$ ($F(1,46) = 8.48$, MSE = 2198224.44).

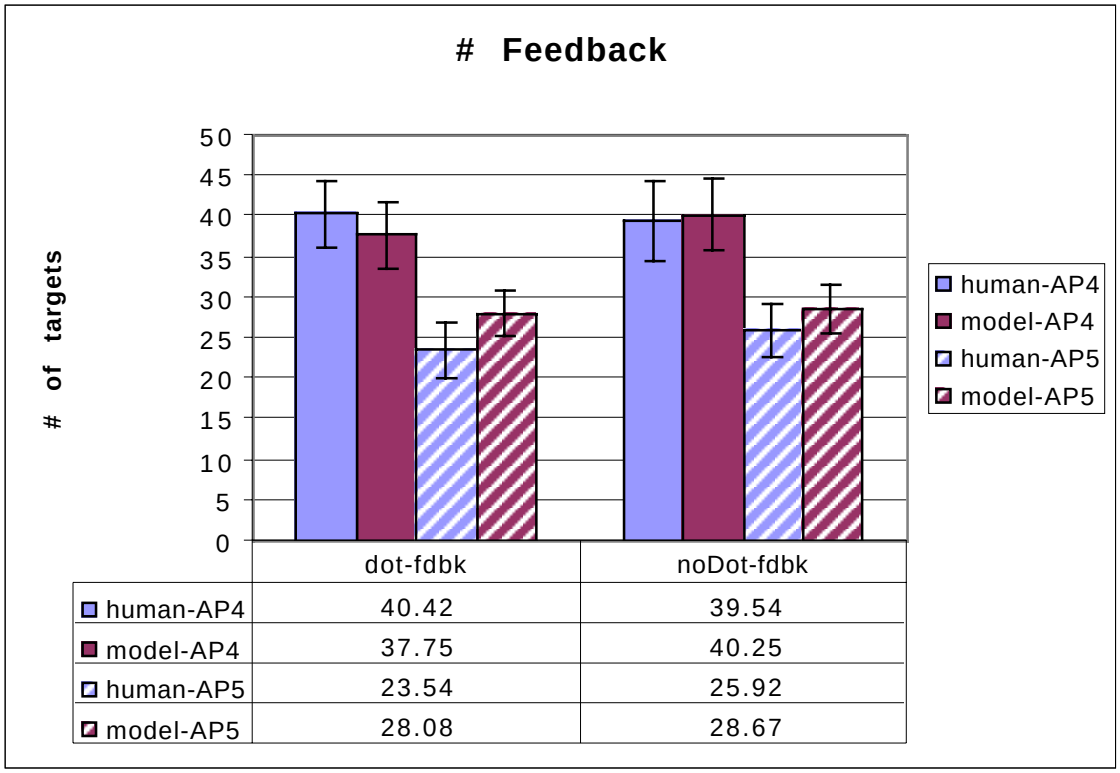
6.3.3. Feedback Processing Results

This section discusses the results for the Feedback Processing unit task. The main quantity result is the total number of times that the feedback was used. The number of feedback track number accesses is given. The time measurement is the mean time spent in Feedback Processing.

6.3.3.1. Number of Feedback Accesses

This measure indicated the number of times in the immediate feedback condition that feedback was accessed by the subject. After a classification is entered into the system the immediate feedback window is covered with a gray window. To access the feedback, the subject must click on the gray window. This measure only applies to the feedback condition.

Figure 31 shows that in AP4, the fit of the Argus model to the human data is good. The ANOVA reports $p = 0.87$ ($F(1,46) = 0.026$, $MSE = 890.97$). Neither the dot-noDot nor the subject type by dot-noDot is significant. The power analysis resulted in a power of 0.053, a small effect size of 0.023, and 15,234 subjects would be required to produce a significant result.



	dot-fdbk	noDot-fdbk
human-AP4	40.42	39.54
model-AP4	37.75	40.25
human-AP5	23.54	25.92
model-AP5	28.08	28.67

Figure 31. Mean number of times the immediate feedback window was accessed during the course of a scenario.

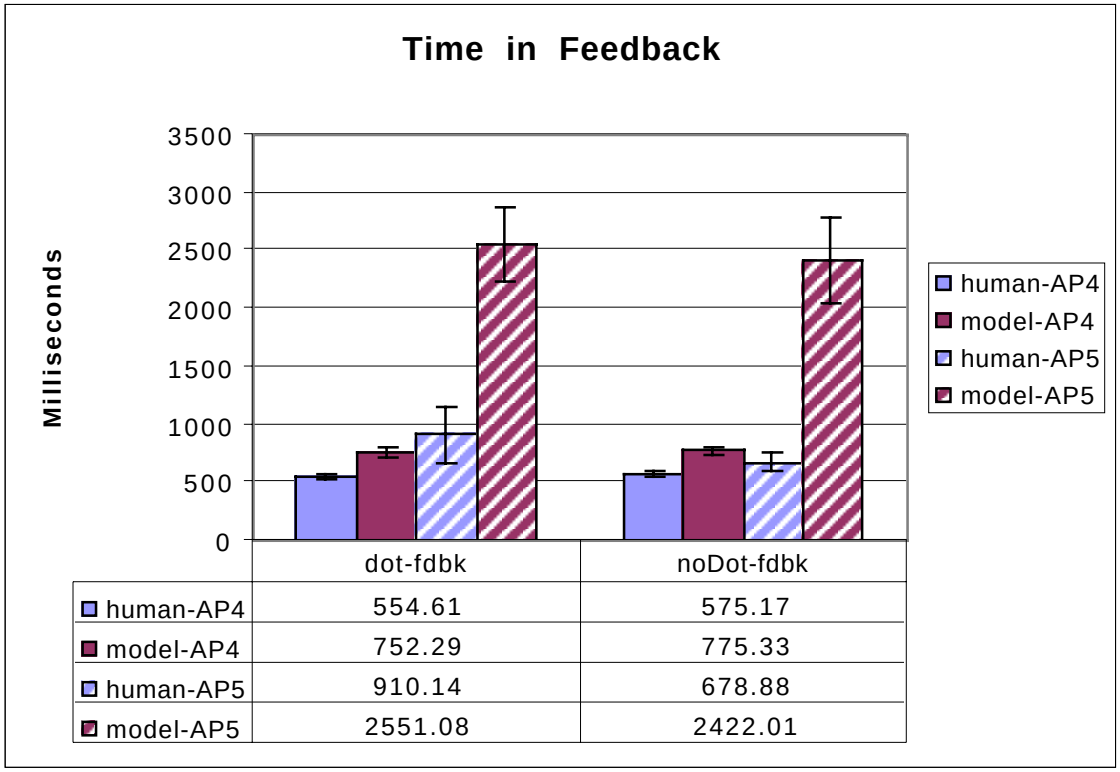
In AP5, the fit of the Argus model to the human data is not quite as good as in AP4. The ANOVA reports p = 0.4 (F(1,46) = 0.71, MSE = 448.31). Neither the dot-noDot nor the subject type by dot-noDot is significant. The power analysis resulted in a power of 0.13, a small effect size of 0.12, and 548 subjects would be required to produce a significant result.

6.3.3.2. Time In Feedback Processing

The total time spent in feedback processing is the time from when the gray window is clicked to expose the feedback information to the time that the cursor leaves the feedback window and it is covered over again with the gray window. Since this measure is circumscribed by mouse movements, it is only an upper bound of the feedback processing time for a human. It is

possible to leave the cursor over the window while engaging the perceptual and cognitive systems to perform target searches.

In AP4, Figure 32 shows significant differences between the Argus model and human data. The ANOVA reports, $p = 0.0003$ ($F(1,42) = 15.46$, $MSE = 56309.83$). Neither the dot-noDot or the subject type by dot-noDot is significant.



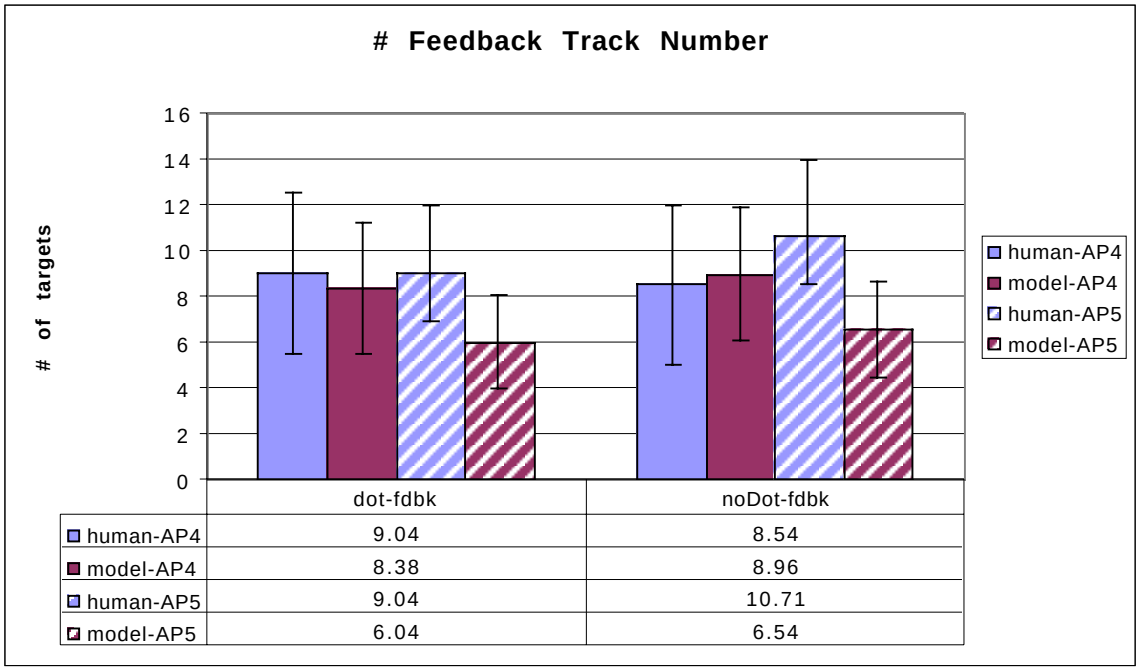
	dot-fdbk	noDot-fdbk
human-AP4	554.61	575.17
model-AP4	752.29	775.33
human-AP5	910.14	678.88
model-AP5	2551.08	2422.01

Figure 32. Mean time in milliseconds spent in feedback processing. The time from clicking on the gray window to uncover the feedback to the cursor leaves the window, recovering it.

In AP5, like AP4, there are significant differences between the Argus model and human data. The ANOVA reports $p < 0.0001$ ($F(1,35) = 21.25$, $MSE = 2447094.75$). Neither the dot-noDot or the subject type by dot-noDot is significant.

6.3.3.3. Number of track number accesses

The feedback window contains the track number of the target being processed. The track number is covered with a gray box which must be clicked on to uncover the track number. Thus there is an additional cost in terms of motor processing to use the track number. The benefit of the track number is that it provides information on the target that was classified. The intent from an interface design view is to reduce the number of reclassifications of already classified targets.



	dot-fdbk	noDot-fdbk
human-AP4	9.04	8.54
model-AP4	8.38	8.96
human-AP5	9.04	10.71
model-AP5	6.04	6.54

Figure 33. Mean number of times the feedback track number was uncovered.

Figure 33 shows that in AP4, the fit of the Argus model to the human data is good. The ANOVA reports $p = 0.98$ ($F(1,46) = 0.001$, $MSE = 484.73$). Neither the dot-noDot nor the subject type by dot-noDot is significant. The power analysis resulted in a power of 0.05, a small effect size of 0.004, and 490,560 subjects would be required to produce a significant result.

In AP5, the fit of the Argus model to the human data is not quite as good as in AP4. The ANOVA reports p = 0.34 (F(1,46) = 0.92, MSE = 336.42). The main effect of dot-noDot was significant, p = 0.039 (F(1,46) = 4.54, MSE = 6.21). The interaction of subject type by dot-noDot is not significant. The power analysis resulted in a power of 0.16, a small effect size of 0.14, and 404 subjects would be required to produce a significant result.

6.3.4. Tracking Task

The measures presented in this section are the overall tracking score, the RMS error, the time in tracking and the number of switches to the Tracking unit task from the Classification Task.

6.3.4.1. Tracking Score

Tracking score is a measure of how well the subject tracked the plane over the course of the entire scenario.

Tracking score = (# of tracking cycles in normal state / Total # of tracking cycles) * 100

A tracking cycle is 5 seconds which is how often the Root Mean Square Error is calculated and the test is performed to determine if the error is below threshold. If it is below threshold it is in the normal state. The results presented are the mean scores for all four scenarios. This score was continuously displayed to the subject on the status bar. The instructions for the tracking task directed the subject to try keep this score equal to the classification task score.

Figure 34 shows that the Argus model and human subjects did about the same, p = 0.64 (F(1,46) = 0.22, MSE = 762.97). No interactions involving subject type were significant. The power analysis yielded a power of 0.073, a small effect size of 0.065, and 1,850 subjects would have been required to produce a significant result. Although the fit is good, the variability in the human data was greater that the variability in the Argus model data.

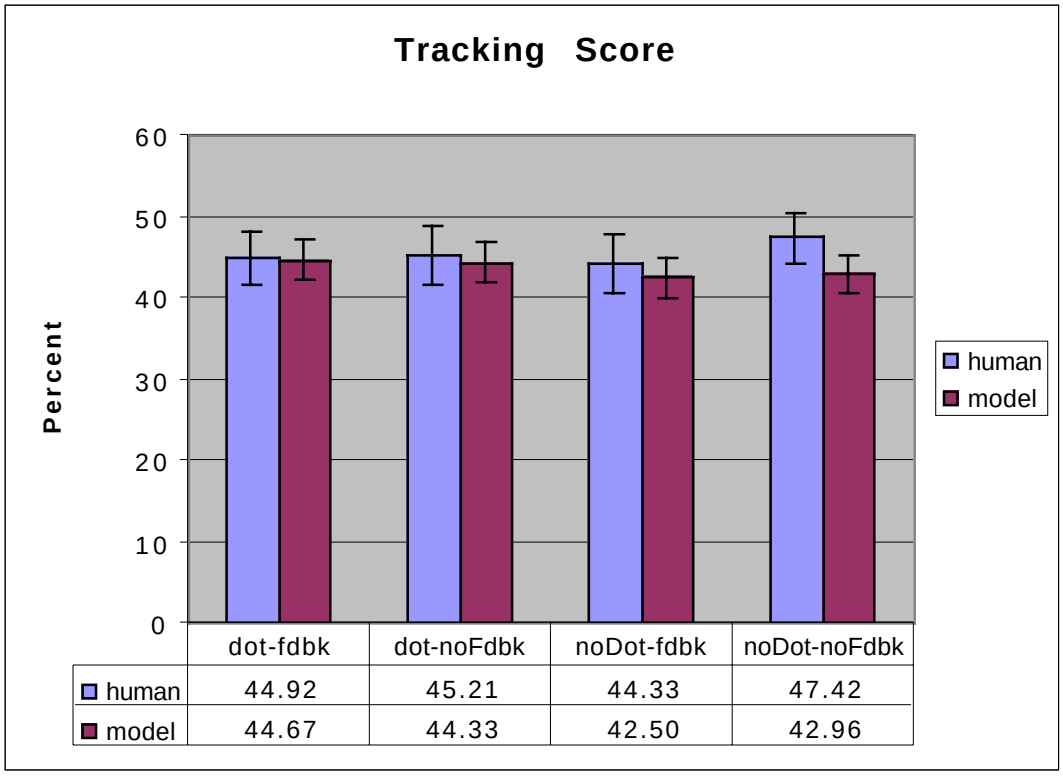


Figure 34. Mean final tracking score for a scenario. % of total of tracking cycles that were within the threshold.

6.3.4.2. Root Mean Square Error

The Root Mean Square Error is a measure of how well the subject maintained the pointing device over the target plane. It is the difference in position of the center of the pointing device and the center of the plane in pixels. If the subject is performing the primary task, the position of the pointing device is its last position before the switch to the classification task from the tracking task.

Figure 35 shows that the Argus model and human subjects did about the same, $p = 0.59$ ($F(1,46) = 0.30$, $MSE = 917.694$). No interactions involving subject type were significant. The power analysis yielded a power of 0.092, a small effect size of 0.089, and 1,006 subjects would have been required to produce a significant result. Although the fit is good, the variability in the human data was greater that the variability in the Argus model data.

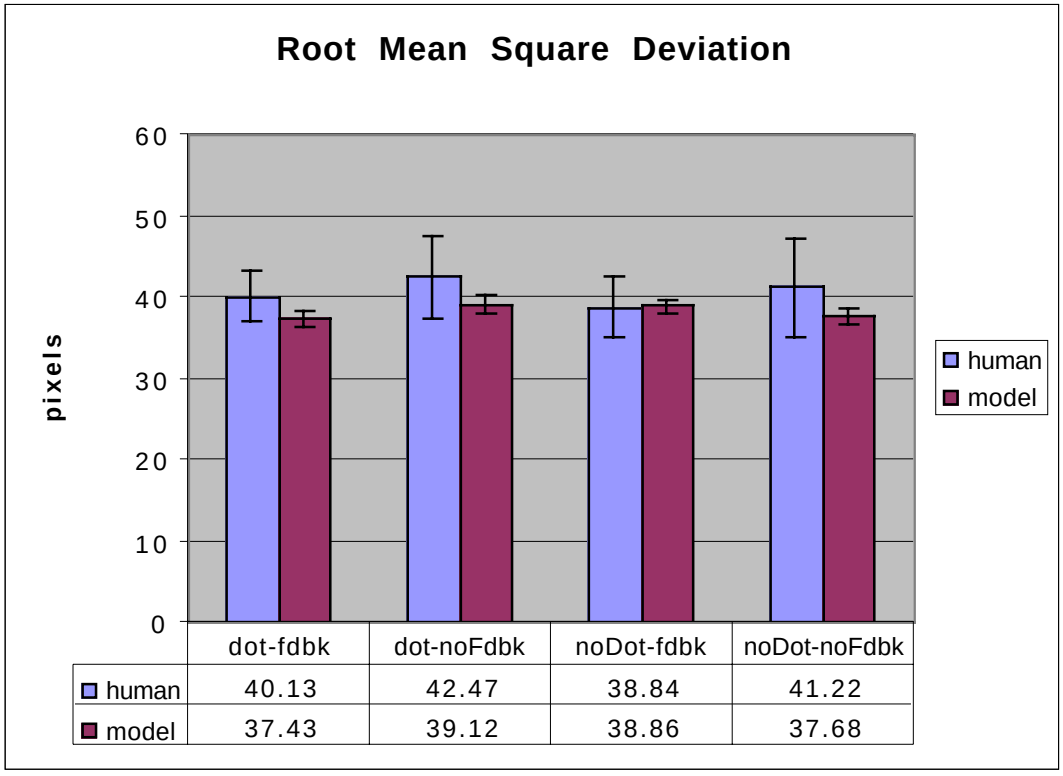


Figure 35. Mean root mean square error in pixels over the course of a scenario.

6.3.4.3. Time in tracking task

The mean total time spent during a 720 second scenario is given in Figure 36. This is the time from when the tracking cursor was activated to the time when it changed back to the normal cursor.

Figure 36 shows that the Argus model spent longer in the tracking task than humans but the difference is not significant, $p = 0.71$ ($F(1,46) = 0.14$, $MSE = 37606.28$). The main effect of feedback was significant, $p = 0.031$ ($F(1,46) = 4.95$, $MSE = 1524.32$). No interactions involving subject type were significant. The power analysis yielded a power of 0.065, a small effect size of 0.053, and 2,798 subjects would have been required to produce a significant result. The variability in the human data is slightly greater that the variability in the Argus model data.

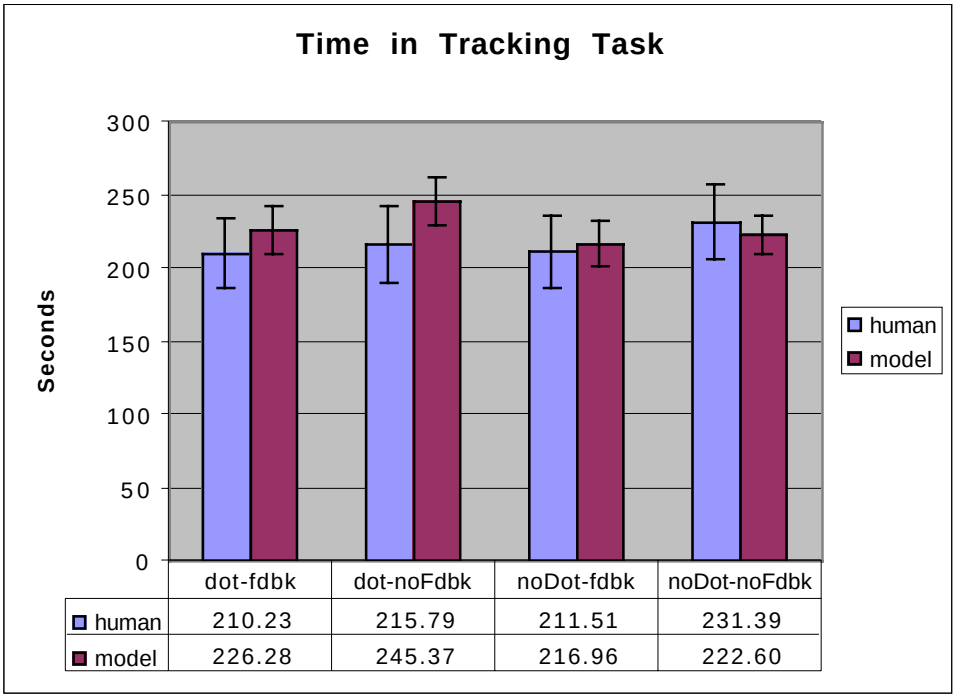


Figure 36. Mean time in seconds spent doing the tracking task.

6.3.4.4. Number of Switches

The Number of Switches is the number of times that execution was switched from doing one task to doing the other task. Figure 37 indicates the human subjects switch significantly more often than the model (SHUs), $p = 0.013$ ($F(1,46) = 6.77$, $MSE = 2411.82$). No interactions involving subject type were significant. The power analysis calculated a power of 0.75, a medium to large effect size = 0.39, and 56 subjects would have been required to achieve a significant result.

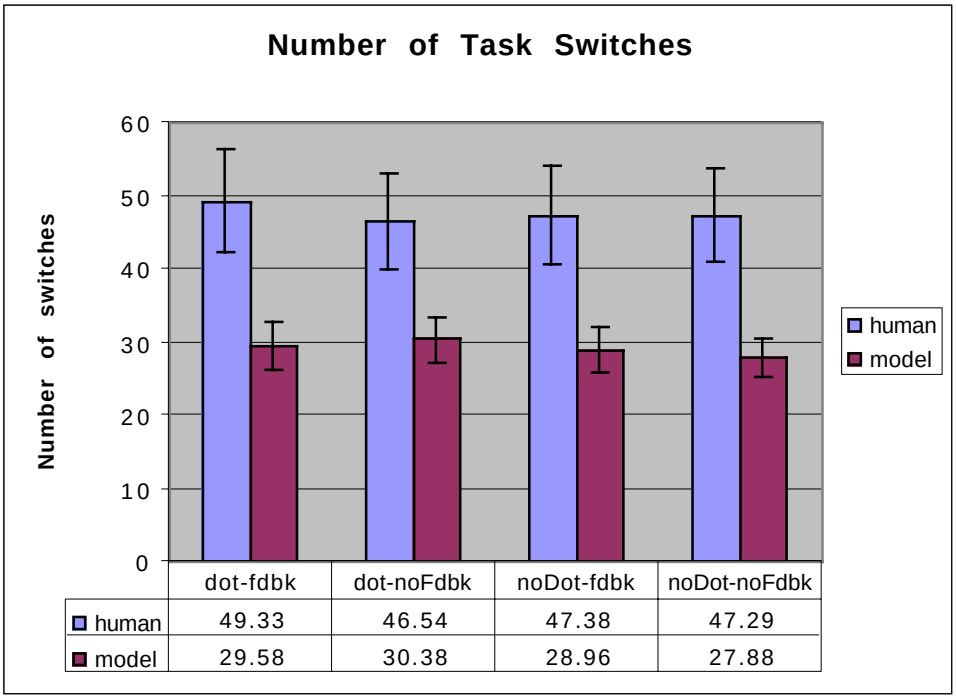


Figure 37. Mean number of times a switch was made from the primary task to the secondary task.

The Argus model only monitors the status of the tracking task at unit task boundaries. The vision module does not have the capability to react to salient changes in the environment. The changing of the tracking cursor to yellow or red may be triggering task switches for human subjects immediately. They would be able to track for a short period of time in order to return the tracking state to normal and switch back to the classification task.

6.4. Summary of Results

The fit of the Argus model to the human data at the task level was excellent. For the duration measures, the AP4 results support the inference that the Argus model performance matches human performance on the Classification task. The only measure for which there was a significant difference of subject type was the time in feedback. Two measures, Initial Incorrect and Total Reclassifications, indicated marginal differences by subject type. The validation is necessary to ensure that any performance difference when the Tracking task is added is not due to a deficiency of the Argus model on the Classification task.

The AP5 results indicate that the Argus model was able to successfully predict the impact of the secondary tracking task on performance of the classification task. Two measures, Feedback Time and Initial Incorrect, indicated significant differences by subject type. The final model resulted from adding knowledge of how to perform the tracking task to a validated model of the primary task.

For the tracking unit task, the results show the only measure on which there was a significant difference was the number of switches.

7. Discussion

The engineering products of this dissertation are the Argus model, instantiating the model as SHUs, and the Argus STE. The combination of SHUs and a STE can be used as a research tool by the HCI community for research on interactive behavior in the judgment and decision making domain. The Argus model has theoretical implications the ACT-R 5.0 cognitive architecture, human variability, decomposing interactive behavior, visual search in dynamic environments, and models of multitasking environments.

7.1. ACT-R 5.0 Cognitive Architecture

The majority of the 100 published ACT-R models are models of laboratory tasks. The future of ACT-R, however, lies in being able to scale up to real world environments in domains such as HCI. This trend is evidenced by the models of simulated task environments such as the Kanfer-Ackerman model (Lee and Byrne 1999), the AMBR model (Lebiere, Anderson et al. 2001) and the UAV model (Gluck, Ball et al. 2002). The Argus model is a level above these models in complexity. Its success shows that the ACT-R 5.0 cognitive architecture is a viable architecture for HCI.

The Argus model made extensive use of the buffer architecture for interleaving cognition, perception and motor actives. The asynchronous operation of these processes is crucial to the ability of the Argus model to meet the time pressures of the task. For example, an interface object can be accessed faster if the motor operation to move the mouse can proceed in parallel with the movement of visual attention.

The sub-symbolic layer of ACT-R is vital to the model's ability to capture within subject variation. The expected gain noise incorporated into the production utility calculation was the mechanism used by the Argus model to capture the way in which humans use interface features. For example, even though feedback is provided during the course of a scenario, a human would not necessarily access it each time a classification was made. The activation noise for declarative memory elements contributed to the ability of the Argus model to simulate the human's pattern of selecting already classified targets.

The activation memory system in ACT-R allows the Argus model to capture memory capacity differences between subjects. During training, all subjects were exposed to the same material. However, they accessed the help facility of the interface in different degrees. This implies they differed in how well they remembered the declarative information presented during training. The Argus model was able to capture this by setting the base levels of the relevant declarative elements to different values for different model subjects.

While the success of the Argus model validates the use of ACT-R, it also unveils some deficiencies of the architecture. In order to achieve reasonable retrieval times for the target declarative memory elements the scaling parameter for the retrieval time equation was set in the Argus model to a value 20 times less than the default. The default was based on the retrieval equation used in ACT-R 4.0. ACT-R 5.0 uses a new memory retrieval time equation, called competitive latency. The equation increases the retrieval time based on the degree of difference in activation of the most active element and its competitors. The Argus model shows that if the number of competitors is large as is the case with targets in the Argus STE, as time progresses and all targets are being accessed about the same number of times, then the retrieval times increase to an unreasonable number (e.g. 10 seconds and greater).

The Argus model demonstrated the need for a bottom up or stimulus driven visual attention mechanism. The Argus model performed poorly in simulating the number of switches from the primary task to the secondary task. The Argus model polls the visual memory for the stimulus (i.e. the tracking icon would change color to indicate that the tracking task needed attention) at unit task boundaries. A stimulus driven mechanism would enable the detection of the color change more often than the polling mechanism. A third alternative, that was not tried, is to have a production that detects the stimulus and competes for execution on every cycle.

7.2. Modeling Human Variability

The success of the Argus model in matching the behavior of human users can be attributed to the ability of the Argus model to capture the range of human interactive behavior. Variation in interactive behavior occurs within a single user and between users. As discussed above, the ACT-R 5.0 sub-symbolic layer provides mechanisms to simulate the within subject variation due to the architecture.

In complex interactive task environments such as the Argus STE, the between human variability emerges from the combinatorial properties of the ETA triad. Humans with different knowledge and experience interact with interfaces that afford a variety of actions to do tasks that have relatively few constraints. In this dissertation, the approach which is taken to account for the between subject variability is to create a SHU. A SHU is the Argus model instantiated with a given behavioral profile. The behavioral profile specifies how this particular instantiation of the Argus model will react to the task and interface conditions. One objection to this approach might be that it increases the free parameter space of the model and models should have a minimum number of free parameters. The counter argument is that these parameters reflect differences in

experience and knowledge of humans and therefore have a meaning at a different level than architectural parameters. The architectural parameters are indeed fixed across each SHU.

Human variability in complex interactive tasks has implications for development of a priori engineering models. Kieras (Kieras, Wood et al. 1995) states that for a model to be truly predictive, different combinations of modeling policies should be implemented based on a task analysis. This dissertation shows that this is necessary but not sufficient. The within subject and between subject variation that occurs in unconstrained environments requires that an approach that produces a range of behavior is also required to predict aggregate performance.

7.3. Decomposing Interactive Behavior

No ACT-R model has been compared to human behavior on as many output measures as the Argus model. The intent of this detailed analysis is twofold. First, an accurate simulation of the actions that combine to produce behavior at the unit task level provides an explanation for unit task level behavior. This explanation strengthens the predictive power. Second, an inaccurate simulation identifies weak points in the Argus model and therefore identifies areas where further investigation is required to understand the human behavior. The success of this approach will benefit other models based on task decomposition, such as the Kanfer-Ackerman and the UAV models described earlier.

7.4. Modeling Visual Search In Dynamic Environments

Visual search in the Argus STE differs from visual search in laboratory environments in that the task constraints, the design of the interface, and embodied cognition interact to produce the pattern of target selection. For example, the task constraint of classifying an unclassified target before selecting the next target imposes an interruption of the search process of a few

hundred milliseconds (target was classified) to tens of seconds (target was unclassified and must be classified). This interacts with the memory capacity to recall target locations and declarative information about if the target has been classified. The interaction is actually a three way interaction because the interface design in the dot condition provides information about the classification status of the target. This information can be accessed by a perceptual and a motor action, which impose a cost that must be weighed against memory retrieval costs. The task constraints, the memory/perceptual/motor constraints, and design factors can combine in varying degrees at different times. The problem from a modeling point of view is how to capture these weighted dynamic interactions.

The analysis of the human eyetracking data indicates that most humans have a tendency to perform some sort of systematic search. The search strategies incorporated into the Argus model were the ones identified by playing back the human data and identifying these systematic patterns. In order to perform this organized rather than random search it is necessary to remember the location (to some degree of accuracy) of the last target. The Argus model gives this location information special status by storing it in the goal so that it can be accessed without retrieval.

When humans move their eyes back to the radar screen after classifying a target they analyze the visual scene (the physical layout of the targets on the screen) in terms of the task constraint that each target must be classified in each sector. The modeling issue is how to capture this scene analysis capability. The solution presented here is to build a representation of the environment which contains task relevant information about the entire scene. For example, one piece of information represented is the number of targets within one degree of visual angle of the border. This scene representation is then used by a special function that determines the utility of each search strategy built into the model.

If a search strategy is successful then visual attention is moved to the location returned by the search strategy to encode the track number at that location. The task constraint of determining the classification status of the target interacts with interface design. If the interface provides a visual aid, perceptual and motor processes can be used or the track number can serve as a cue to recall the classification status. If the interface does not contain a visual aid then memory trace is the only option. The data indicates that humans are sensitive to features of the interface especially if it reduces cognitive effort. The classification time data also indicates that if no perceptual/motor option is available, more effort (because of task requirements) is put into encoding the declarative fact that the target is being classified. The Argus model captures this by the production utility and declarative activation mechanisms. The perceptual/motor initiating productions are given greater utility than the memory rehearsal productions in the visual aid (dot) condition. In the no visual aid condition (noDot), the Argus model performs more rehearsals of the classification state of the target than in the visual aid condition.

7.5. Models of Multitasking Environments

In the Argus STE, the Classification task is a task that requires a mix of cognitive, perceptual and motor operations. The Select Target and Feedback Processing unit tasks are mostly perceptual and motor tasks, while the Classify Target unit task is mostly cognitive. The added task is a perceptual motor task but does require some cognitive processing to monitor the tracking score to maintain a balance with the primary task. In the ACT-R 5.0 architecture all perceptual and motor operations are initiated by productions, therefore, moving the mouse to maintain the tracking icon over the plane requires production execution. The human data indicated that humans switched from the primary task to the secondary task at unit task boundaries. The Argus model captured this by having productions that monitored the color of the

tracking icon compete with productions that initiated the next unit task. If the color of the tracking icon indicated that the tracking task needed attention then the tracking unit task would be initiated rather than the unit task of the primary task. The weak point of the Argus model was that the number of times the model switched from the primary to secondary task was less than the number of times humans switched. The human data indicates that humans, for the most part, do switch on unit task boundaries rather than in the middle of a unit task. The fact that they switch more times than the Argus model implies that they switch even when the feedback indicates the switch is not necessary.

8. CONCLUSION

The Argus model presented in this dissertation predicts human performance over a span of twelve minutes in a multitask environment under varying interface conditions. The Argus model is the first ACT-R 5.0 model to simulate human performance in such an environment. This effort represents a significant step in establishing ACT-R 5.0 as “the” cognitive architecture for HCI research in non-laboratory environments. However, the Argus model also identifies directions for the on-going development of the ACT-R theory and implementation. The most important architectural deficiency suffered by the Argus model was the lack of a stimulus driven attentional mechanism. A second deficiency is the lack of a scene analysis capability. The solution implemented in this model is to extend the architecture with a conflict resolution function that constructs a representation of the environment. This function fulfills the needs of this dissertation but is domain specific. A more general solution is required.

A road block to the acceptance of computational cognitive modeling as a tool in behavioral research is that rule-based models have only been able to simulate a small amount of human variability. The Argus model achieves a much greater degree of variability due to its SHU approach. While this approach was successful, the initial settings for the subjects were ad hoc and a more formal methodology for accomplishing this needs to be investigated.

The state of the art in predictive cognitive modeling is to predict task performance. The combinatorial complexities of interactive behavior however make it possible for a number of different combinations of actions to achieve the same task result. A model could get task performance right by being wrong on the actions that compose the task. In interface development

the designer often is concerned how adding, deleting, or changing some interface mechanism affects the series of actions required to do the task. Therefore it is important for the Argus model to provide a veridical simulation below the task level. The approach taken with the Argus model is to decompose and analyze the model data at a level below the task level. When the analysis shows differences between the model and human behavior it points to ways to improve the model.

The ultimate goal for this line of research is to provide user interface designers with a tool that improves the quality of interfaces and reduces development costs. Interface quality will be improved if design decisions are made on predictions grounded in embodied cognition rather than the ad hoc manner that is typically the norm today. The Argus model demonstrates that such predictions can be made and are robust to changes in interface conditions.

SHUs can serve as avatars of real users during all phases of interface development reducing time and costs incurred in testing with real users. The Argus ACT-R 5.0 model demonstrates the feasibility of computation cognitive models as simulated users. The Argus model simulates an interactive user on several levels. It directly interfaces with the target system through the same hardware and software as a real user. It can be instantiated with the interface and task strategies of different users to generate aggregate data. These different users could be all expert users, all novice users or a mixture. It could be instantiated as one particular user to address individual differences in interactive behavior.

Several significant improvements are being planned. First, the cognitive model should learn interface strategies, rather than requiring the interface engineer to supply the necessary knowledge. Second, like interface knowledge, knowledge of the task must be supplied by the engineer. The model would be more general if it could obtain task knowledge by reading and interpreting instructions. Third, the model performs many repetitive actions (e.g. such as adding

numbers or pressing buttons) that are implemented in productions that are specific to the model. These productions could be generalized and packaged in a library for use in other domains by other modelers. This dissertation has advanced the engineering of an embodied cognitive model as an interface design tool, but further research is needed to make this an even more useful tool for an interface design engineer.

Bibliography

Bibliography

Adelman, L., E. D. Henderson, et al. (2001). Vicarious functioning in teams. The essential Brunswik: Beginnings, explications, and applications. K. R. Hammond and T. R. Stewart. New York, Oxford University Press: 416-422.

Adelman, L., S. L. Miller, et al. (in press). "Using Brunswikian Theory and a Longitudinal Design to Study How Hierarchical Teams Adapt to Increasing Levels of Time Pressure." Acta Psychologica.

Altmann, E. M. and W. D. Gray (2000). "Functional decay: A memory model of task switching and maintenance." Manuscript submitted for publication.

Altmann, E. M. and W. D. Gray (2000). An integrated model of set shifting and maintenance. Proceedings of the Third International Conference on Cognitive Modeling. N. Taatgen and J. Aasman. Veenendal, NL, Universal Press: 17-24.

Anderson, J. R. (1990). The adaptive character of thought. Hillsdale, NJ, Erlbaum.

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

Anderson, J. R. (2001). Introduction to ACT-R 5.0. ACT-R Post Graduate Summer School 2001.

Anderson, J. R. and C. Lebiere, Eds. (1998). Atomic components of thought. Hillsdale, NJ, Erlbaum.

Anderson, J. R., M. Matessa, et al. (1997). "ACT-R: A theory of higher-level cognition and its relation to visual attention." Human-Computer Interaction 12(4): 439-462.

Ballard, D. H., M. M. Hayhoe, et al. (1995). "Memory representations in natural tasks." Journal of Cognitive Neuroscience 7(1): 66-80.

Ballard, D. H., M. M. Hayhoe, et al. (1997). "Deictic codes for the embodiment of cognition." Behavioral and Brain Sciences 20(4): 723-742.

Ballas, J. A., C. L. Heitmeyer, et al. (1992). Evaluating Two Aspects of Direct Manipulation in Advanced Cockpits. Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems: 127-134.

Brehmer, B. and D. Dörner (1993). “Experiments with computer-simulated microworlds: Escaping both the narrow straits of the laboratory and the deep blue sea of the field study.” Computers in Human Behavior 9(2-3): 171-184.

Byrne, M. D. (1999). ACT-R Perceptual-Motor (ACT-R/PM): A users manual, http://chil.rice.edu/byrne/RPM/docs/index.html.

Byrne, M. D. (2001). “ACT-R/PM and menu selection: applying a cognitive architecture to HCI.” International Journal of Human-Computer Studies 55: 41-84.

Byrne, M. D. (in press). Cognitive Architecture. Handbook of Humcan-Computer Interaction. J. Jacko and A. Sears. Hillsdale, NJ, Erlbaum.

Byrne, M. D. and J. R. Anderson (1998). Perception and action. The atomic Components of Thought. J. R. Anderson and C. Lebiére. Hillsdale, NJ, Erlbaum: 167-200.

Card, S. K., T. P. Moran, et al. (1983). The psychology of human-computer interaction. Hillsdale, NJ, Lawrence Erlbaum Associates.

Chong, R. S. (1998). Modeling single-task performance improvement using EPIC-Soar. Ninth Midwest Artificial Intelligence and Cognitive Science Conference. Menlo Park, CA, AAAI Press: 10-16.

Cohen, J. (1988). Statistical Power Analysis for the behavioral sciences. New York, Academic Press.

Donchin, E. (1995). “Video games as research tools: The Space Fortress game.” Behavior Research Methods, Instruments, & Computers 27(2): 217-223.

Dwyer, D. J., J. K. Hall, et al. (1992). A performance assessment task for examining tactical decision making under stress. Orlando, FL, Naval Training System Center.

Ehret, B. D. (2000). “Learning where to look: The acquisition of location knowledge in display-based interaction.” Dissertation Abstracts International: Section B: the Sciences & Engineering 60(10-B): 5239.

Freed, M. and R. Remington (2000). Making human-machine system simulation a practical engineering tool: An APEX overview. Proceedings of the Third International Conference on Cognitive Modeling. N. Taatgen and J. Aasman. Veenendall, NL, Universal Press.

Gilliland, S. W. and R. S. Landis (1992). “Quality and quantity goals in a complex decision task: strategies and outcomes.” Journal of Applied Psychology 77: 672-681.

Gluck, K., J. Ball, et al. (2002). Modeling Basic Maneuvering with the Predator UAV, The Air Force Office Of Scientific Research. 2002.

Gray, W. D. (2000). “The nature and processing of errors in interactive behavior.” Cognitive Science 24(2): 205-248.

Gray, W. D. (2002). “Simulated task environments: The role of high-fidelity simulations, scaled worlds, synthetic environments, and microworlds in basic and applied cognitive research.” Cognitive Science Quarterly.

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

Gray, W. D., B. E. John, et al. (1993). “Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world performance.” Human-Computer Interaction 8(3): 237-309.

Gray, W. D., M. J. Schoelles, et al. (2000). Modeling a continuous dynamic task. Proceedings of the Third International Conference on Cognitive Modeling. N. Taatgen and J. Aasman. Veenendal, The Netherlands, Universal Press: 158-168.

Hollenbeck, J. R., D. R. Ilgen, et al. (1995). “Mutlilevel theory of team decision making: Decision performance in teams incorporating distributed expertise.” Journal of Applied Psychology 80(2): 292-316.

Hollenbeck, J. R., D. J. Sego, et al. (1997). Team decision making accuracy under difficult conditions: Construct validation of potential manipulations using the TIDE 2 simulation. Team performance, assessment, and measurement: Theory, research, and applications. M. T. Brannick, E. Salas and C. Prince. Hillsdale, NJ, Erlbaum.

Hornof, A. J. (2001). Predictice engineering models of interface usability: cognitive modeling and human-computer interaction. ACM Special Interest Group on Computer & Human Interaction. 2002.

Hornof, A. J. and D. E. Kieras (1997). Cognitive modeling reveals menu search is both random and systematic. ACM CHI'97 Conference on Human Factors in Computing Systems. New York, ACM: 107-114.

Johnson, T. R., J. Krems, et al. (1994). A computational model of human abductive skill and its acquistion. Sixteenth Annual Conference of the Cognitive Science Society: 463-468.

Just, M. A. and P. A. Carpenter (1992). “A capacity theory of comprehension: Individual differences in working memory.” Psychological Review 99: 122-149.

Kieras, D. E. and D. E. Meyer (1996). The EPIC architecture: Principles of operation. Ann Arbor, University of Michigan.

Kieras, D. E. and D. E. Meyer (1997). “An overview of the EPIC architecture for cognition and performance with application to human-computer interaction.” Human-Computer Interaction 12(4): 391-438.

Kieras, D. E., S. D. Wood, et al. (1995). Predictive engineering models using the EPIC architecture for a high-performance task. Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. New York, ACM Press: 11-18.

Kieras, D. E., S. D. Wood, et al. (1997). "Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task." ACM Transactions on Computer-Human Interaction 4(3): 230-275.

Lebiere, C. (1998). ACT-R 4.0: A User's Manual. Pittsburg, PA, Carnegie Mellon University.

Lebiere, C., J. R. Anderson, et al. (2001). Multi-tasking and cognitive workload in an ACT-R model of simplified air traffic control. Proceedings of the Tenth Conference on Computer Generated Forces and Behavior Representation, Norfolk, VA.

Lee, F. J. and J. R. Anderson (2000). Modeling eye-movements of skilled performance in a dynamic task. Proceedings of the Third International Conference on Cognitive Modeling. N. Taatgen and J. Aasman. Veenendal, NL, Universal Press.

Lee, F. J. and M. D. Byrne (1999). Modeling Dynamic tasks: Implications for ACT-R/PM. 6th Annual ACT-R Workshop, George Mason University.

Meyer, D. E. and D. E. Kieras (1997). "A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms." Psychological Review 104(1): 3-65.

Myers, B., S. Hudson, et al. (2000). "Past, present, and future of user interface software tools." ACM Transactions on Computer-Human Interaction 7(1): 3-28.

Newell, A. (1973). You can't play 20 questions with nature and win: projective comments on the papers of this symposium. Visual Inforamation Processing. W. G. Chase. New York, Academic Press.

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

Newell, A., P. S. Rosenbloom, et al. (1989). Symbolic architectures for cognition. Foundations of Cognitive Science. M. I. Posner. Cambridge, MA, MIT Press: 93-131.

Newell, A. and H. A. Simon (1963). GPS, a program that simulates human thought. Computers and Thought. E. A. Feigenbaum and J. Feldman. Cambridge, MA, MIT Press: 279-293.

Pylyshyn, Z. W. (1984). Computation and Cognition: Toward a Foundation for Cognitive Science. Cambridge, MA, MIT Press.

Pylyshyn, Z. W. (1991). The role of cognitive architecture in theories of cognition. Architectures for Intelligence. K. VanLehn. Hillsdale, Lawrence Erlbaum Associates, Inc.

Ritter, F. and R. Young (2001). "Embodied models as simulated users: introduction to this special issue on using cognitive models to improve interface design." International Journal of Human-Computer Studies 55: 1-14.

Ritter, F. E., G. D. Baxter, et al. (2000). “Supporting cognitive models as users.” ACM Transactions on Computer-Human Interaction 7(2): 141-173.

Salvucci, D. D. (2000). A model of eye movements and visual attention. Proceedings of the Third International Conference on Cognitive Modeling. N. Taatgen and J. Aasman. Veenendal, NL, Universal Press: 252-259.

Salvucci, D. D. (2001). “Predicting the effects of an in-car interface on driver performance: an integrated model approach.” International Journal of Human-Computer Studies 55: 85-107.

Salvucci, D. D. and K. L. Macuga (2001). Predicting the effects of cell-phone dialing on driver performance. Proceedings of the 2001 Fourth International Conference on Cognitive Modeling. E. M. Altmann, A. Cleeremans, C. D. Schunn and W. D. Gray. Manwah, NJ, Lawerence Erlbaum Associates.

Schoelles, M. J. and W. D. Gray (2001). “Argus: A suite of tools for research in complex cognition.” Behavior Research Methods, Instruments, & Computers 33(2): 130-140.

Taatgen, N. (1999). Learning without limits: from problem solving toward a unified theory of learning. Psychology. The Netherlands, University of Groningen.

Triesman, A. and G. Gelade (1980). “A feature integration theory of attention.” Cognitive Psychology 12: 97-136.

Wickens, C. D. (1992). Engineering psychology and human performance. New York, HarperCollins.

Young, R. M., T. R. G. Green, et al. (1989). Programmable user models for predictive evaluation of interface designs. ACM CHI'89 Conference on Human Factors in Computing Systems. New York, ACM Press: 15-19.

Young, R. M. and J. Whittington (1990). Using a knowledge analysis to predict conceptual errors in text-editor usage. ACM CHI'90 Conference on Human Factors in Computing Systems. J. C. Chew and J. Whiteside. New York, ACM Press: 91-97.