Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL: Universal Press.

Argus Prime: Modeling Emergent Microstrategies in a Complex, Simulated Task Environment

Michael J. Schoelles & Wayne D. Gray
Human Factors & Applied Cognition
George Mason University
Fairfax, VA 22030 USA
+1 703 993 2104
mschoell@gmu.edu

ABSTRACT

Cognition, perception, and motor actions weave a tangled web. At times, the three may proceed independently of each other. At other times, they may have complex, sequential dependencies such as, for example, when the decision to click on a button waits for perception to return to cognition the information that the cursor and button objects are co-located. Indeed, much of what is commonly called *cognitive workload* may have less to do with cognition and more to do with an interface-induced tangle of cognition with perception with motor actions. In this paper, we describe a simulated task environment (Gray, in preparation), Argus Prime, and an approach to modeling the tangled web of interactions induced by its interface.

Keywords

ACT-R/PM, microstrategies, simulated task environment, attention, motor movement, computational cognitive models, interactive behavior

INTRODUCTION

Dynamic decision making tasks often require periods of intense cognitive, perceptual, and motor activities. These activities can be combined in a finite number of ways to accomplish a variety of subtasks. We refer to these combinations as microstrategies (Gray & Boehm-Davis, 1999; Gray, Schoelles, & Fu, 1999). Different microstrategies may require different amounts of effort to execute. Likewise, the effort involved in any given microstrategy may be unequally partitioned among that microstrategy's cognitive, perceptual, and motor components.

With all else equal, the microstrategy that entails the least-effort should be deployed. We assume that the least-effect judgements of microstrategies are not made in isolation, but in the context of the constraints and opportunities provided by cognition, the task, and the artifact used to perform the task (Gray, in press; Gray & Altmann, in press). Hence, in computer-based simulated task environments (Gray, in preparation), the least-effort microstrategy will vary as a function of task and interface. In the work we describe below, the task is fixed and we pursue the general question of how characteristics of the interface can induce selection of those microstrategies that lead to the best performance.

Modeling that combines cognitive, perceptual, and motor actions has been called embodied cognition (Kieras & Meyer, 1997). In real-world tasks the three types of actions can proceed in parallel but with a complex web of sequential interdependencies. Models of embodied cognition can quantitatively measure the effort required by each interactive component to determine the optimal division of labor. Often, albeit not always, such embodied models directly perceive and manipulate the interface that is used by human users. This feature cleanly separates the model from the interface (Ritter, Baxter, & Jones, 2000) and can provide an effective means for interface designers to evaluate new designs or design changes.

This paper reports on the use of ACT-R/PM (Byrne, 1998) to study embodied cognition. Our goal is to understand how subtle aspects of an interface may lead to large increases in cognitive workload. Our models are situated in the Argus Prime simulated task environment. Argus Prime presents subjects with a task reminiscent of that performed by radar operators. The general task is to classify targets on a 7-point threat scale.

The task involves four subtasks. For target selection, the user attends to icons on the screen (perception), decides to process an icon (cognition), and selects it (motor). In information retrieval the user reads the raw data values for this object (perception). Score calculation entails mapping raw data to target score (cognition), mapping score to threat value (cognition), selecting a threat value (perception and motor), and entering the decision (motor). Finally, feedback processing consists of perceiving feedback (perception) and processing the feedback (cognition). As this brief task analysis illustrates, each subtask combines cognitive, perceptual, and motor operators. Less apparent from this overview is the complex web into which the three types of actions for each subtask are woven.

The ACT-R/PM architecture combines ACT-R's theory of cognition (Anderson & Lebiére, 1998) with modal theories of visual attention (Anderson, Matessa, & Lebiére, 1997) and motor movement (Kieras & Meyer, 1997). ACT-R/PM explicitly specifies timing information for all three processes as well as parallelism between them. The software architecture facilitates extensions

Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL: Universal Press.

beyond the modal theory of visual attention and motor movements.

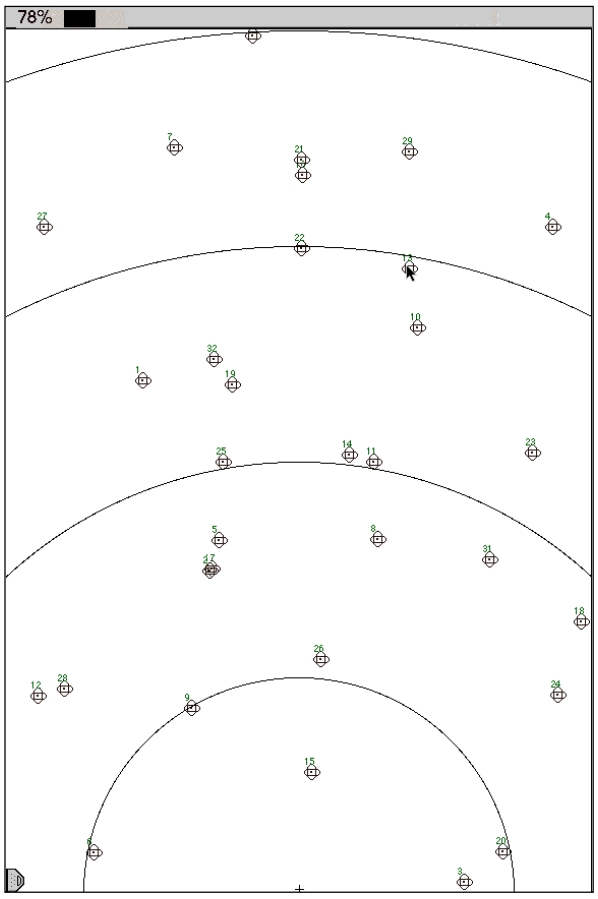

Figure 1: The left-half of the Argus Prime display. This half shows the radar screen and status bar (top). Ownship is indicated by the small "+" at the bottom.

To describe our modeling work it is first necessary to discuss the tasks that can be performed with Argus Prime and to show the interface with which the model and human subjects interact. We then briefly describe several empirical studies. With this background in place the model and its interactions with the system are presented, along with the predictions from the model for the current study. We conclude with a discussion of possible extensions to ACT-R/PM.

ARGUS PRIME: TASK AND SIMULATED TASK ENVIRONMENT

In Argus Prime the user's goal is to correctly assess the threat value of each target in each sector of a radar screen. As shown in Figure 1, the screen represents a portion of an airborne radar console with 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 velocity 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.

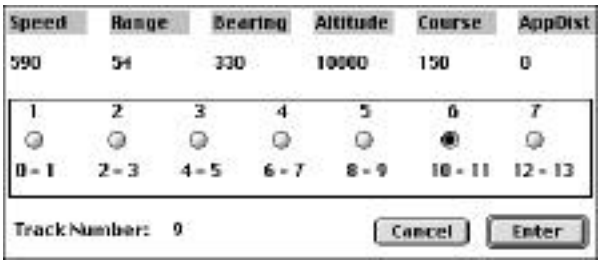

Figure 2: Argus Prime information window for Target 9. The figure shows the target's current speed in mph, range in miles, absolute bearing in degrees, altitude in feet, course in degrees, and the current estimate of closest approach distance in miles. Below the attribute information are radio buttons used for entering the estimated threat value of the target on a 1-7 scale. Subjects are taught an algorithm that combines the raw information to yield a target score between 0-13. The target score is then mapped onto the 7-point threat scale. The black dot in radio button #6 indicates the threat value assigned this target by the subject.

The user selects a target by moving the cursor to its icon and clicking. When a target has been selected, the information window (shown in Figure 2) appears in the right half of the screen (to the right of the radar shown in Figure 1). 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. The bearing is the orientation of the target from ownship in degrees (0 -360). Range is the distance in miles (0 - 200) from ownship. Speed is the speed of the target in miles per hour (0 - 5000). 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 user must convert the attribute values to cue values that collapse the attribute values to a single number (0 - 3 with 0 being the least threatening). For example, if the range of a target from ownship is 165 miles, its cue value for range is 0 - low threat. If the range from ownship is 37 miles, its cue value for range is 3 - high threat.

The score for a target is a weighted combination of cue values of a subset of the attributes. In the experiments we have conducted the relevant attributes are range, altitude, and approach distance with weights 2, 1, and 2 respectively. (Hence, a cue value of 3 for range would be multiplied by 2 to yield a weighted value of 6.) Help for

Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL: Universal Press.

each attribute is available by clicking on the attribute labels shown in gray in Figure 2

Figure 2 shows that the target score is mapped to a 7-point threat value by the user selecting the radio button that corresponds to the threat value. This threat value is inputted to the computer by clicking on enter button (bottom right of Figure 2). If the user correctly classifies the target, full credit is given. If the user 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 user can classify it, it is considered incorrectly classified and a score of zero is assigned.

Feedback to the user takes two forms both of which appear on the left hand side of the status bar shown at the top of Figure 1. Summative feedback 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 given for each classification decision made by the user. The colored box in the status bar indicates a correct decision (black) or an incorrect decision (orange). Colored feedback (gray) is also given if the user attempts to enter the same score for the same target twice in one sector.

A log file is created for each scenario completed by a user. The file contains each user 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.

EMPIRICAL STUDIES AND MICROSTRATEGIES

At the current writing, Argus Prime has been used in two completed empirical studies and a pilot study has been conducted as preparation for a third study. The studies vary from three to five hours in length. The first hour is devoted to training and the last hour to collecting individual difference data on working memory capacity. The remaining hour(s) is devoted to four or more 12-15 min Argus Prime scenarios.

The details of the three studies and their data analyses comprise a rich and interesting story, but one that will not be told here. Instead, our goal is to motivate our modeling work by focusing on a few important interface variations and providing qualitative descriptions of several microstrategies that our subjects used. Our plan is not to enumerate the empirical findings, but to provide enough details so that the modeling challenge will be appreciated.

An analysis of the first study revealed that some subjects used a memory-less microstrategy (Ballard, Hayhoe, & Pelz, 1995) during the transition from the target selection to the information retrieval subtasks. Specifically, the link between a target icon and the information window is the target number. This number is displayed above the target and in the lower left corner of the information window Figure 2). For the Look-After microstrategy, the subject encodes the target number, clicks on the target's icon, looks at the information window, and verifies that the current information represents the intended target by matching the encoded number with the number displayed in the information window. The alternative microstrategy, Look-Before, is simply to place the cursor over the intended target, look at the information window, then click on the icon and notice that all of the information in the window changes at the same instant. Analysis of the eye data has shown that subjects follow both these strategies.

For the first two studies, 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 the threat value radio button 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 2) when the target was reselected. We call this combination of no change to the target's on-screen icon and persistence of the classification in the information window the *noChange-dotOn* interface.

The third study, currently in the pilot phase, manipulates the ease of retrieving history information ("have I classified this target already") from the display. In addition to noChange-dotOn, two new interfaces will be used. The *noChange-dotOff* interface is similar to the noChange-dotOn in that the target's on-screen icon does not change when a classification is made. It differs from noChange-dotOn in that the information window contains no record (i.e., the dot is off) as to whether a target has already been classified. In contrast, for the *Change-dotOff* interface the on-screen icon for targets that have been classified changes color. 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 pattern of behavior suggests that for the noChange-dotOn 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 is unclear in these studies whether this memory-less strategy is adopted by choice or whether, under the

Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL : Universal Press.

conditions of the study, human cognition is incapable of retrieving target status information. This issue will be tested empirically and analytically by the data and models built for the third study.

Performance on the noChange-dotOn interface will be used as a baseline with which to compare the other conditions. We expect the noChange-dotOff interface to force the memory versus microstrategies issue. If subjects have no memory for having classified a target, they will be required to waste time recomputing the algorithm to reclassify already classified targets. In contrast, the Change-dotOff condition provides a memory-less way to avoid classified targets and to focus on unclassified ones. Hence, subtle changes in the interface will enable different sets of microstrategies between the three conditions. These different microstrategies are expected to be differentially successful and to result in large differences in performance.

THE MODEL

Interface to task

The ACT-R/PM code executing the model runs as a separate process from Argus Prime. This process starts when the scenario starts. All communication between the model and Argus Prime is through the motor and vision module commands of ACT-R/PM. Functionally, the interface of the model to Argus Prime is the same as the human to computer interface.

Strategy Implementation

For each subtask, the microstrategies that we have identified have been built into the model. At present the model neither acquires nor adapts microstrategies by itself. Rather, we are using ACT-R/PM in model tracing mode (see, for example, Anderson, Boyle, Corbett, & Lewis, 1990; Gray, in press) to explore the performance implications of various combinations of microstrategies. For example, if we believe we have identified the microstrategies that subject AP1234 regularly uses for target selection, information retrieval, score calculation, and feedback processing, we can select these microstrategies from the *model configuration dialog box* (see Figure 3). The model will perform the scenario using the candidate combination of microstrategies. After performance, we can compare the model's log file with the subject's log file to determine patterns of similarity and differences.

Each microstrategy of a subtask contains matching goal chunks so that all the microstrategies for the same subtask are in the conflict set. ACT-R calculates expected gain dynamically and, when alternative productions exist, it chooses the production with the highest expected gain (see Anderson and Lebiere, 1999). However, to match strategy to subject, we use the probability of success parameter, r, to set the expected gain for each production at the beginning of the scenario. For example, if for the target hooking subtask a subject used the Look-After microstrategy, its r-value would be set to 1.0 and the r-value for Look-Before to 0.01.

Figure 3: Model configuration dialog box.

Model Components

The structure of the model parallels the four subtasks: target selection, information retrieval, classification, and feedback processing.

Target Selection

There are two stages to target selection, the initial stage in which no targets have been classified and the maintenance phase in which targets are reclassified after they cross into a new sector or a new target appears. Thus, cognition is faced with two problems: keeping tracking of what has been classified and directing perception to notice new targets or targets crossing boundaries.

Ten productions implement the target selection task. Microstrategies have been identified for the search phase and for the transition from target icon to target information.

The model uses two features of ACT-R/PM to select targets. In the initial and maintenance phase a systematic search is carried out by a scan production. This production uses ACT-R/PM's *Find Location* command to create a visual location with top-down and bottom-up constraints. Top-down constraints determine the area in which to search. The bottom-up constraint is that the location must contain the feature *target-feature*. Cognition can direct the search for target-features in a systematic fashion by changing the search area. If a target-feature is found then an episodic, declarative memory element (or chunk) of the visual location is created. At this point, perception has delivered to

Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL: Universal Press.

cognition something that cognition can retrieve and process further. For the next step cognition sends the *Move Attention* command to the vision module with the visual location created by Find Location as an argument. After attention is moved, the vision module creates an episodic trace of the visual object it finds at that location.

The systematic direction of search in Argus Prime is based on our preliminary analysis that many subjects move within sectors from right to left and between sectors from closest to ownship to farther away. The model achieves this behavior by moving the scan area. When and how to interrupt this systematic search to process a new target or one that is about to cross a border is an open question. A mechanism is required to determine when the target icon and black arc are close to intersecting. Issues that must be resolved include whether two feature searches are required for each target (one for the target and one for a black arc) and whether two Move Attentions are required or if a single Move Attention command should return both objects?

Since the empirical studies suggest that users have no memory for prior classifications, the model does not try to retrieve a previous visual object (i.e., episodic trace). This implies that the model does not have to rehearse any traces. Thus, this scanning strategy has a low cognitive but a high perceptual cost.

For the Change-dotOff condition the interface provides a memory of target classification by changing color when classified and returning to its original color when it needs to be classified again. In this case, the model executes a production that adds a color constraint to the Find Location command. This *Popout* microstrategy searches fewer locations than the scan strategy with the result that fewer location and object episodic traces are created.

After the Move Attention command creates an episodic trace of the target, the model sends a command to the motor module to move the cursor to the target. ACT-R/PM uses Fitts Law to calculate the time for cursor movements. While the cursor is moving, the target could move off the screen. The model verifies that the cursor and button are collocated through a fail production that has a low utility. The fail production fires only in the case that the cursor movement has completed and there is no target at the cursor's current location.

If the cursor and target are collocated, one of two microstrategies may be implemented. As discussed above, in the Look-Before microstrategy the eyes are moved to the information window before the target icon is clicked. In the Look-After microstrategy the icon is clicked, its target identification number is noted, and then the eyes are moved.

Information Retrieval

Twenty-two productions retrieve the target attribute data or use help, if necessary, to retrieve cue value information from the display instead of from memory.

The model interacts with the information window to process the target attribute data. For the range attribute two possibilities exist. The range can be determined spatially by encoding the radar screen sector in which the icon appears, or it can be read textually from the information window.

To process the attribute data contained by the information window the model sets a goal with slots for each of the relevant attributes. Find Location commands are executed for each attribute. When the location trace indicates that text for the attribute exists at this location and has not been attended, attention is moved to the location to create a text visual object. The model then creates a semantic trace of the real number from the text visual object. To map the semantic representation to a cue value the model retrieves, using partial matching, memory elements whose slots are attribute name, cue value, and the midpoint of the range of values for this cue value. For example, if the actual range is 127 miles, partial matching will retrieve 125 miles, the midpoint for range cue value 2.

We do not think that this solution is satisfactory as it relies on partial matching which requires that similarities are set between the raw value and the midpoint. Although it seems reasonable that such similarities should be learned, ACT-R/PM does not have a learning mechanism for doing so.

At present the model sets the similarity between the mapping chunk and the number chunk when it is created from the text. This solution assumes that the user fills in the midpoint slot when he or she learns the mappings during training. An approach that is being explored is to replace the midpoint slot with the low and high values of the mapping. If a mapping chunk based on the raw value cannot be retrieved, the model would use a compute strategy by searching through the mapping chunks using less than and greater than operators to determine if the raw value fit into the specified range. An episodic chunk would be created containing the raw value, the cue value, and attribute name. This chunk would be rehearsed, so that future retrieval would be successful.

If the mapping chunks are below activation, the model uses help. A Find Location is done to locate the label on the information window of the attribute being processed. Attention and the cursor are moved to that location and the label is clicked. When the interface displays the help screen, the set of productions to process the help screen is executed to calculate the cue value.

Each cue value is weighted. The model initially contains memory element chunks with two slots, name and weight, for each attribute. If one of these chunks is below threshold, the help sequence will be invoked. The model uses multiplication facts to multiply the weight times the cue value.

Classification

After the weighted values are calculated, they must be summed, and the total score mapped to the 7-point threat value scale. The mathematics and mapping required for classification are implemented by fourteen productions.

The model adds the cue values together in two steps using addition facts. First, the range and altitude values are added. Second, this sum is added to the approach distance value. This total represents the target score. As seen in Figure 2, the information window contains seven radio buttons one of which must be selected to register the classification. Below each button is text containing the low and high target score. The model looks in this area of the screen to encode the location of the text that includes the target score number. The model uses this location to encode the location of the corresponding push button. Now a sequence of productions is executed that encodes the push button visual object, moves the cursor to it, and clicks on it. Finally, to complete the classification task the model locates, attends to, moves to, and clicks on the enter button.

Feedback

The model processes the feedback given by the system by executing a find location constrained by the coordinates of feedback area of the display and color. Three productions compete for execution, one for each possible color.

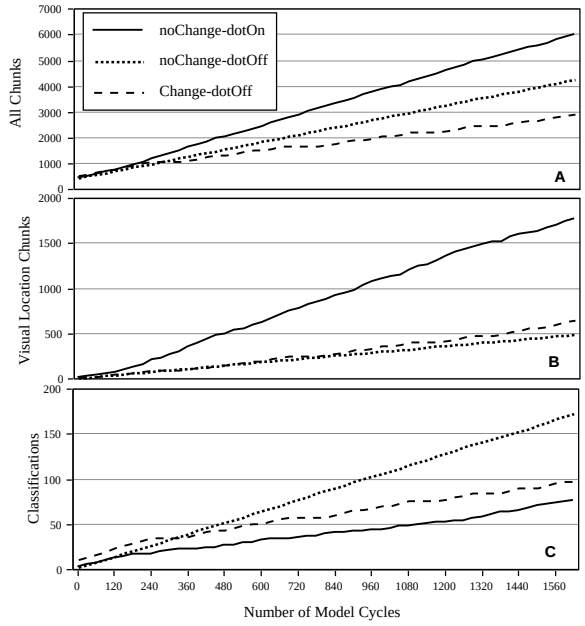

Figure 4: A-C. Model results over a run of 1600 cycles as a function of three alternative interface designs for displaying (or not displaying) the classification status of a target (see text for discussion). (A) shows the total number of all chunks (declarative memory elements) created. (B) indicates the total number of visual location chunks created. (C) shows the number of target classifications that were calculated and entered.

Data Collection

In addition to the log file that is created for a human subject, a second log is created for the model that contains information about the contents of declarative memory such as the total number of chunks, number of visual locations, number of visual objects, etc.

Initial Results

As a prediction for experiment 3, the model was run holding all microstrategies and productions constant except those for the three proposed interface conditions -- noChange-dotOn, noChange-dotOff, and Change-dotOff. Figure 4 shows the difference between strategies for the number of all chunk types created (Figure 4-A), the visual location chunks created (Figure 4-B), and the targets classified (Figure 4-C) for each of the three interface conditions.

Figure 4-A shows that through cycle 240, all conditions creates an equal number of chunks of all kinds. When a scenario begins, the subject is confronted with 18 unclassified targets. The interface-specific microstrategies do not differ in the way in which they handle this period. After all initial targets are classified, the Change-noDot interface results in fewer chunks created as only targets that are not classified are attended to. In contrast, the noChange-dotOff condition creates fewer total chunks than the noChange-dotOn condition. This difference reflects the fact that noChange-dotOff spends more time than noChange-dotOn recalculating the classification algorithm for already classified targets. Consequently, noChange-dotOn scans more visual locations and checks on more information windows than the other conditions.

Figure 4-B shows the number of visual location chunks created. As for the total number of chunks, the noChange-dotOn condition creates the largest number of chunks that encode a visual location. Interestingly enough, the graph shows little difference between the noChange-dotOff and the Change-dotOff condition. In the former case, so much time is being spent calculating algorithms that few targets can be attended. In the latter case, the model simply does nothing unless the feature detection process returns an unclassified target (as indicated by its color).

Figure 4-C completes the picture. Because it classifies every target which it attends, the noChange-dotOff condition makes the most classifications. Unfortunately, most of these are wasted effort as the targets have already been classified (our current model assumes no memory for classifications). The contrast between Change-dotOff and noChange-dotOn is interesting. The former classifies more targets than the latter as it wastes no time inspecting a target unless its feature detection process has indicated that it is unclassified. The latter looks at many more targets (see Figure 4-B) but most of these are already

Schoelles, M. J., & Gray, W. D. (2000). Argus Prime: Modeling emergent microstrategies in a complex simulated task environment. In N. Taatgen & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling (pp. 260-270). Veenendal, NL: Universal Press.

classified. Hence, there are a certain number of unclassified targets that it misses as it is distracted by having to move to, click on, and double check (but not reclassify) targets that are already classified.

These results are intriguing and suggestive; however, it is important to remember their limitations. For the three interface conditions all microstrategies were the same except for those directly relevant to the differences in the interface. Hence, the variations between-subjects has not been captured. For each interface, we have only implemented one microstrategy. For example, the only microstrategy for noChange-dotOff is the memory-less one.

At present the data from experiments 1 and 2 are being analyzed to discover other microstrategies and, specifically, to look for individual differences in the combinations of microstrategies used. The results of this effort are feeding directly into our on-going efforts to model more microstrategies. By the time the data for experiment 3 are collected, we hope to have families of microstrategies that span the gamut of those used by our subjects.

ACT-R/PM Extensions

While ACT-R/PM provides a good foundation for modeling microstrategies, the initial Argus experiments have raised a set of detailed research questions that sugggest expansion of ACT-R/PM's modal model of visual attention. In this paper, we provide a simple list with a short discussion of these issues; however, each issue is the subject of analytic and empirical research in the Argus effort.

- The long-standing debate in the visual attention community between object-based and location-based visual attention is relevant to the Argus task (for example, see Cave & Bichot, 1999; Logan, 1996).
- Is target selection is totally feature based or totally indexed? Can ACT-R/PM be extended to include visual indexing mechanisms and the capability for move attention to take an index as an argument (Pylyshyn, 1998).
- At present, ACT-R/PM has a theory of visual attention but not of eye movements. The analysis of the initial eye tracking data shows many more eye movements than shifts of attention. To investigate the timing constraints that arise from these actions, an eye movement command is required in ACT-R/PM (Lee & D., 1999 also argued for inclusion of saccades and fixations).
- In Argus Prime, target search is conducted by sweeping right-to-left within a sector, beginning with the bottom sector (closest to ownship) and iterating after sweeping the top sector. Since detection of target objects close to sector lines is critical, the issue of divided attention is brought into play. Currently the ACT-R/PM attention mechanism will only focus attention on a single object.
- The target selection processing described above raises two questions. First, how large an area can be searched in one Find Location command. Second, how close does the actual object need to be to the location specified as the argument to the Move Attention command. This question directly relates to the size of the spotlight in spotlight models of visual attention (Wolfe, 1994). The modal theory underspecifies this issue leaving it as a free parameter for the modeler. Aspects of the modal theory that need to be specified include the size and control of the current focus of attention, time staged creation of features for visual objects, and feature degradation depending on distance from fovea.

Our plans for Argus Prime include incorporating into the target selection task a theory of multiple object tracking. Sears and Pylyshyn (in press) have applied the FINST model (Pylyshyn, 1989) to multiple object tracking. This theory hypothesizes a stimulus driven mechanism that individuates objects in the environment by pointing to them; that is, assigning an index. The indexing precedes object identification and the index remains bound to the object even if characteristics of the object change. In particular, if the location of the object changes continuously then the index can still be used to point to the object. Attention can be directed to the object with the index as its argument. The dynamic environment of Argus Prime seems well suited to modeling this theory as a possible mechanism used by subjects in the target selection phase.

CONCLUSIONS

Computational models of embodied cognition are a step towards realizing Newell's challenge to model the complete task (Newell, 1973). The approach took here has several parts. First, a simulated task environment, Argus Prime, was built of a dynamic, real-time task in which a complex web of cognitive, perceptual, and motor actions were required for task performance. Second, data were collected of subjects using the simulated task environment over an extended period. Third, the data were analyzed to identify the microstrategies used by subjects for each subtask. Finally, production models were built that implement some of the microstrategies used by our subjects.

The work reported here is part of Project Argus. The challenge of Project Argus is to determine how interface features interact with the task and cognition to affect cognitive workload. As our models demonstrate, interactive behavior in complex tasks is constrained not only by cognition but by perception and motor processes as well. Although these constraints are at the millisecond level, the milliseconds added to an interaction matter when the task requires thousands of interactions over an extended period of time.

Further work in Project Argus includes expanding the modal models of visual attention and motor movement as well as working to incorporate a modal model of eye movements. Such expansions are necessary to built models that respond adaptively to subtle differences in interface design. The long-term goal of this effort is to create a model that responds adaptively to any change in the Argus Prime simulated task environment. We hope that the lessons learned from this effort can be exported to other task environments (simulated or real) to provide designers with timely feedback on their designs.

ACKNOWLEDGEMENTS

This work was supported by Air Force Office of Scientific Research Grant # F49620-97-1-0353. We thank the many members of the Argus Group who have contributed to the Argus Prime studies: Erik Altmann, Deborah Boehm-Davis, Susan Schnipke, Ryan Snead, and Marc Todd.

REFERENCES

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence, 42*, 7-49.

Anderson, J. R., & Lebiére, C. (Eds.). (1998). *Atomic components of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Matessa, M., & Lebiére, C. (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., Hayhoe, M. M., & Pelz, J. B. (1995). Memory representations in natural tasks. *Journal of Cognitive Neuroscience, 7*(1), 66-80.

Byrne, M. D. (1998). Perception and action. In J. R. Anderson & C. Lebiére (Eds.), *Atomic components of thought* (pp. 167-200). Hillsdale, NJ: Erlbaum.

Cave, K. R., & Bichot, N. P. (1999). Visuospatial attention: Beyond a spotlight model. *Psychonomic Bulletin & Review, 6*(2), 204-223.

Gray, W. D. (in preparation). Simulated task environments: The role of high-fidelity simulations, scaled worlds, synthetic environments, and microworlds in basic and applied cognitive research. In R. Mahan, D. Serfaty, S. Kirschenbaum, M. McNeese, & L. Elliott (Eds.), *Scaled Worlds (working title)* . Hillsdale, NJ: Erlbaum.

Gray, W. D. (in press). The nature and processing of errors in interactive behavior. *Cognitive Science*.

Gray, W. D., & Altmann, E. M. (in press). Cognitive modeling and human-computer interaction. In W. Karwowski (Ed.), *International encyclopedia of ergonomics and human factors* . New York: Taylor & Francis, Ltd.

Gray, W. D., & Boehm-Davis, D. A. (1999). Milliseconds Matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Manuscript submitted for publication*.

Gray, W. D., Schoelles, M. J., & Fu, W.-t. (1999). Modeling microstrategies in a continuous dynamic task. *Manuscript submitted for publication*.

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

Lee, F. J., & D., B. M. (1999). *Modeling Dynamic tasks: Implications for ACT-R/PM.* Paper presented at the 6th Annual ACT-R Workshop, George Mason University.

Logan, G. D. (1996). The CODE theory of visual attention: an integration of space-based and object-based attention. *Psychological Review, 103*(4), 603-649.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W. G. Chase (Ed.), *Visual information processing* (pp. 283-308). New York: Academic Press.

Pylyshyn, Z. (1998). Visual indexes in spatial vision and imagery. In R. D. Wright (Ed.), *Visual attention* (pp. 215-231). New York: Oxford University Press.

Pylyshyn, Z. W. (1989). The role of location indexes in spatial perception: A sketch of the FINST spatial indexing model. *Cognition, 32*, 65-67.

Ritter, F. E., Baxter, G. D., & Jones, G. (2000). Cognitive models as users. *ACM Transactions on Computer-Human Interaction, in press*.

Sears, C. R., & Pylyshyn, Z. W. (in press). Multiple object tracking and attentional processing. *Canadian Journal of Experimental Psychology*.

Wolfe, J. M. (1994). Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review, 1*, 202-238.