Behavior Research Methods, Instruments, & Computers
2001, 33 (2), 130-140

Argus: A suite of tools for research in complex cognition

MICHAEL J. SCHOELLES and WAYNE D. GRAY
George Mason University, Fairfax, Virginia

Argus simulates a radar-like target classification task. It was developed to support research in measuring and modeling cognitive work load. Argus is used in both single-subject and team modes. However, the Argus system is more than just a simulated task environment. Argus features flexible experimenter control over cognitive work load, as well as extensive data collection and data playback facilities to support the iterative nature of research in complex behaviors. In addition, embodied computational models interact with Argus using the same interface as do human subjects. In this paper, we describe these features, as well as the task simulation. In addition, we describe how the system has been used for experimentation. We conclude with a comparison of Argus with other complex task environments.

Argus is a simulated task environment (Gray, in press) that supports the investigation of cognitive work load in a family of dynamic, time-critical tasks.[1] At its core, Argus requires participants to classify the threat value of multiple targets in a radar-like monitoring task. This task can be performed by individuals or by teams. Details of the basic task are provided in the next section.

Argus was designed after an extensive investigation of similar simulated task environments. The systems investigated include Space Fortress (Donchin, 1995), Advanced Cockpit (Ballas, Heitmeyer, & Perez, 1992), the Team Interactive Decision Exercise for Teams Incorporating Distributed Expertise (TIDE^2) (Hollenbeck et al., 1995; Hollenbeck et al., 1997), and Tandem (Dwyer, Hall, Volpe, & Cannon-Bowers, 1992). Like the Advanced Cockpit and Space Fortress, Argus places a premium on embodied cognition (Kieras & Meyer, 1997) and rapid shifts in serial attention (Altmann & Gray, 2000). Like TIDE^2 and Tandem, Argus emphasizes judgment and decision making in a multiple-cue probability task (see also Gilliland & Landis, 1992). Argus was designed to facilitate the investigation of a broad category of research questions centered on how interface design affects cognitive work load in both team and individual performance.

Beyond the simulation, Argus provides a suite of tools for creating task variations, manipulating experimental design, as well as data collection and analysis. In the next section, we describe the basic Argus task and system and discuss the two basic modes of use—individual versus team. In the following sections, we describe the two parts of the Argus tool suite: the Experimenter Interface, and the Data Output and Analysis tools. We then discuss how Argus has been used in experiments and conclude with a detailed comparison of Argus to similar systems.

This work was supported by Air Force Office of Scientific Research Grant F49620-97-1-0353. Correspondence should be addressed to M. J. Schoelles, Department of Psychology, MS 3F5, George Mason University, Fairfax, VA 22030 (e-mail: mschoell@gmu.edu).

THE ARGUS TASK AND SYSTEM

Argus presents subjects with multiple air-based targets on a radar display. The threat value of each target must be determined before the target enters a new sector. In the scenarios used to date, over the course of a 12- to 15-min scenario, an individual using Argus Prime or a team using Team Argus may classify 70–85 targets.

Hardware Requirements and Software Architecture

The Argus system hardware and software architecture is shown in Figure 1. 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 can be seen in Figure 1, the configuration can support an eyetracker. Our configuration incorporates 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. For Team Argus, both a keyboard and a mouse are used as input devices. Argus Prime requires only a mouse.

The Argus 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,

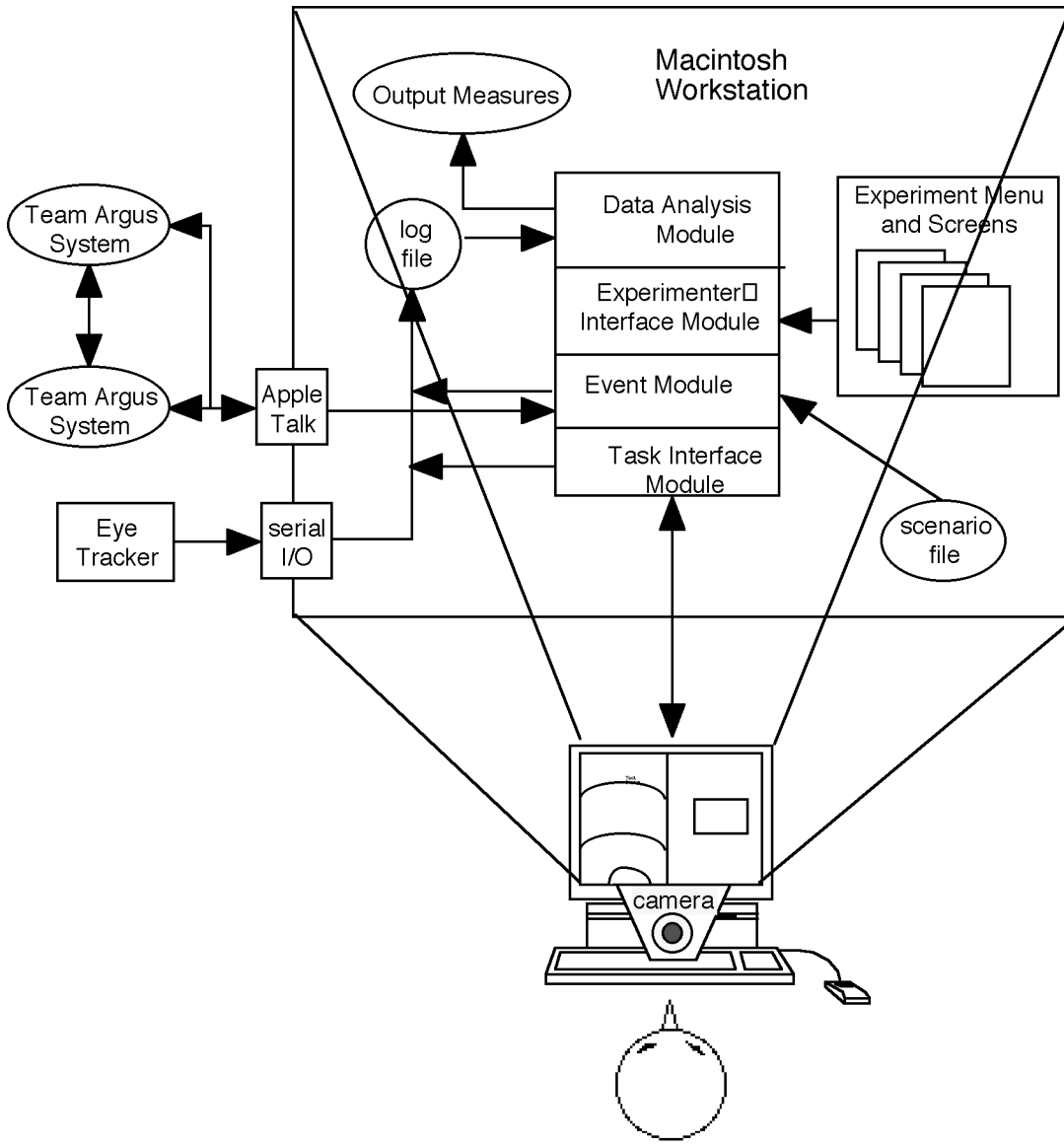


Figure 1. The Argus hardware and software architecture. The eyetracker, Macintosh computer, camera, mouse, and monitor are the hardware components. The Team Argus systems are connected via Appletalk. The software components run the simulation (Event Module and Task Interface Module), interface to the experimenter (Experimenter Interface Module), and analyze the data (Data Analysis Module). Files and interface layouts are shown in the ovals.

as well as all subject actions (i.e., mouse clicks, mouse movements, and eye movements).

Argus Prime and Team Argus Simulation

The Argus task interface depicts an airborne radar console divided into four sectors; each sector is 50 miles wide. Each Argus version has its own radar screen. The left side of Figure 2 shows the radar screen for Argus Prime. The left side of Figure 3 shows the radar screen for Team Argus. The interfaces are different, in part, because Argus Prime can incorporate a secondary tracking task in the right-hand portion of the display screen. The subject is to imagine that his or her position is at the cross at the bottom of the screen in Argus Prime and in the middle of the screen for Team Argus. This position will be referred to as *ownship*.

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, altitude, radar type, and identify friend or foe (IFF) frequency[2] of each target are read from an experimenter-generated file. The scenario can contain events that change a target's speed, course, altitude, radar type, and IFF frequency. For each target, the scenario may contain a single

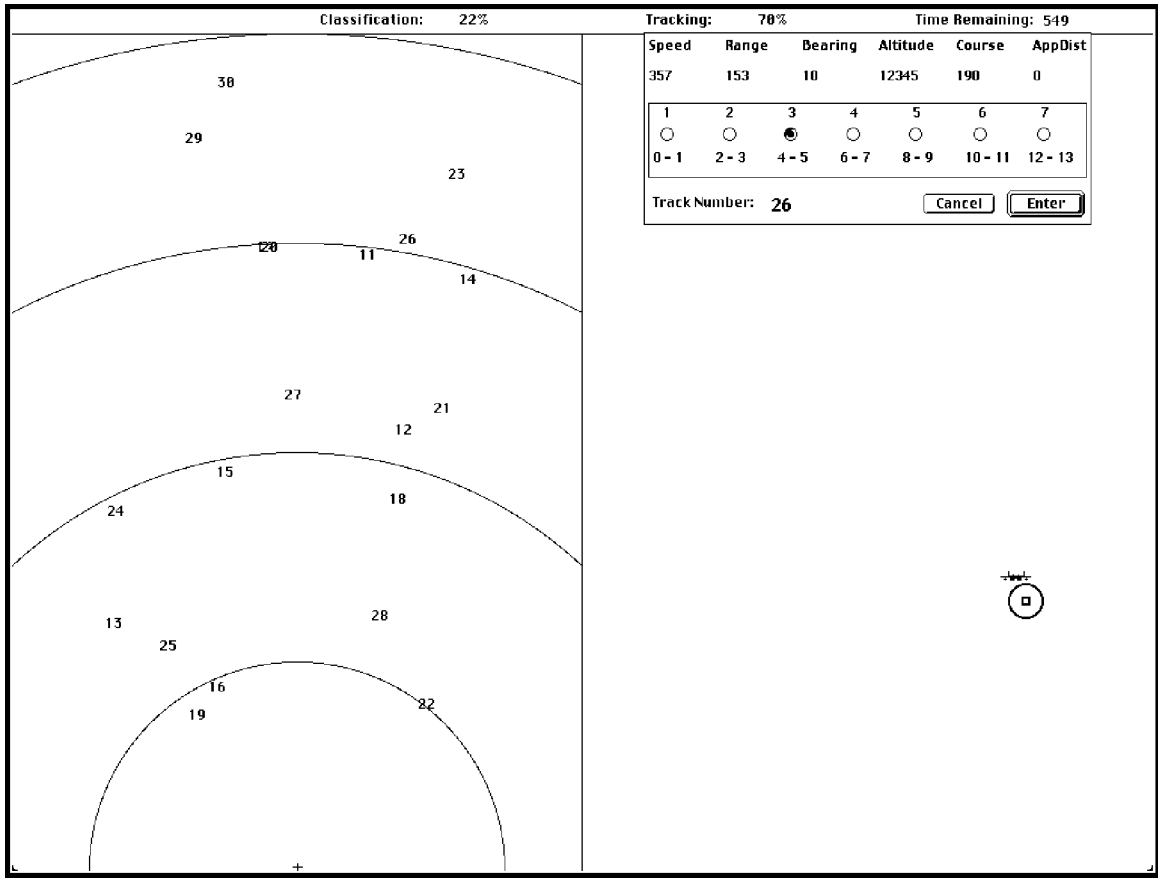


Figure 2. Argus Prime radar screen (left) and information window (upper right). The left half shows the radar screen and status bar (top). Ownship is indicated by the small "+" at the bottom. At the top is the summative feedback shown for classification and tracking tasks, as well as the time remaining 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.

change or multiple changes to any one or all of these attributes at any time. Similarly, new targets can appear at any time during the scenario.

The subject selects (i.e., hooks) a target by moving the cursor to its icon and clicking. Argus can display targets in a variety of ways, some of which are shown in Figure 4. When a target has been hooked, the information about the target appears in a window. Each Argus version has its own information window. The right side of Figure 2 shows the information window for Argus Prime. The right side of Figure 3 shows the Team Argus information window. The Team Argus information window contains interface components that support team communication.

The information window contains the track number of the target hooked 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. Range is the distance in miles from ownship. The bearing is the orientation of the target from ownship in degrees. Altitude is the altitude of the target in feet. Course is the heading of the target in degrees. Approach distance (Argus Prime only) is an estimate of how close in miles the target will come to ownship. Radar is the current state of the radar (off, or, if on, a number indicating its type). IFF is the current state of the IFF (off, or, if on, a number indicating its frequency). The subject is taught an algorithm for combining and converting raw values of the relevant attributes into a threat value. (We provide a detailed example of this algorithm in the Experimenter Interface section.)

One way of manipulating cognitive work load is by creating different Argus games that use different algorithms for computing the threat value of a target. Algorithms can vary in the subset of attributes that they use and in the mapping of raw values to cue values. Similarly, in different algorithms, a given attribute may be weighted differently. (Indeed, the Experimenter Interface Module facilitates changes in the mapping of raw values to cue

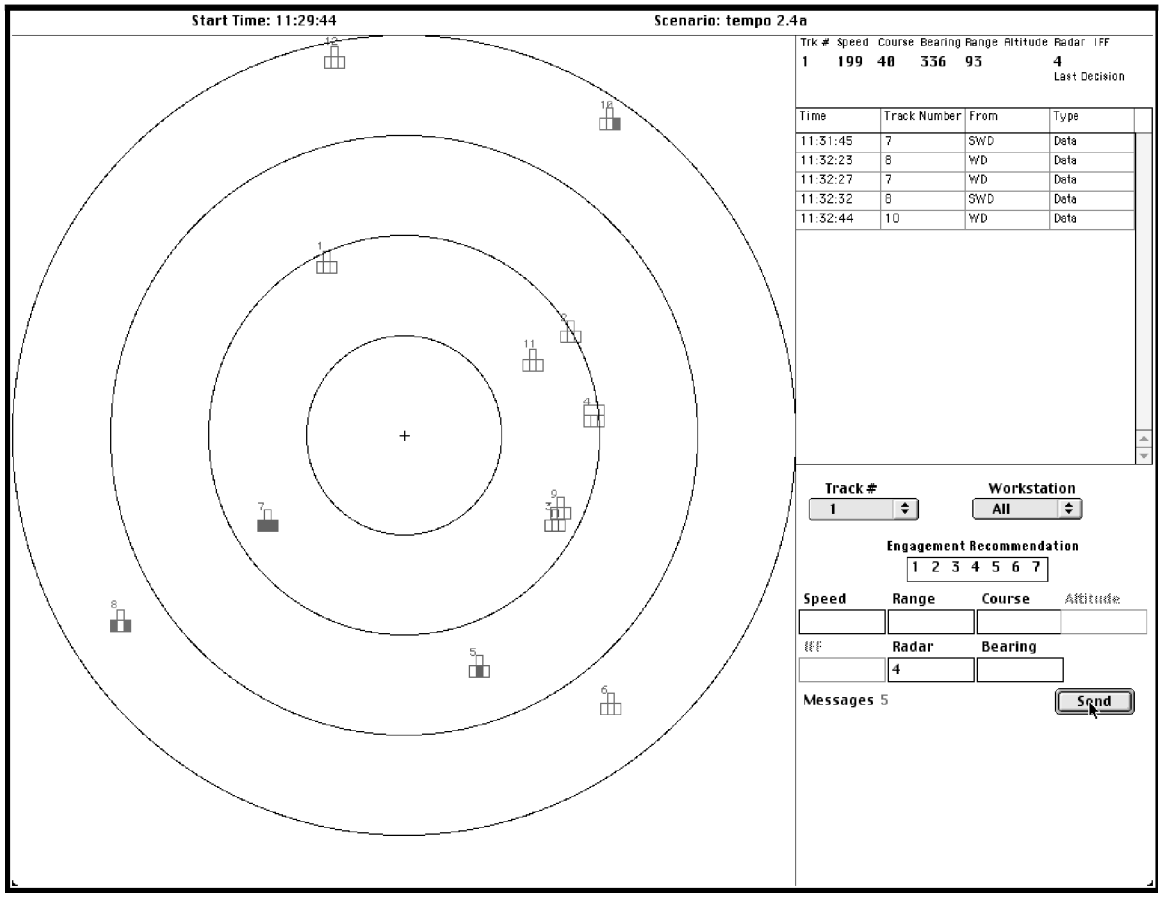


Figure 3. Team Argus radar screen and information window. On the left is the radar screen. Ownship is indicated by the small "+" in the center. The filled-in rectangles in the target icons indicate the sending and receiving of data. The shape of target 4 indicates that the target is 15 sec from the border. The top right side displays the selected target's attributes. Below that is the message queue. The subject accesses this queue to read data sent by the other team members. Below the message queue is the threat value scale and the send facility used to send data to other team members.

values and in the weighting of cue values.) The algorithm yields a score for each target that is mapped to a 7-point threat scale by the subject.

If the subject correctly classifies the target, full credit is given. In Argus Prime, if the subject incorrectly classifies the target, zero points are given. In Team Argus, a partial score is given. Targets must be classified once for each sector that the target enters. If a target leaves a sector before the subject can classify it, it is considered incorrectly classified, and a score of zero is assigned.

In Team Argus, three Argus systems are connected via a local AppleTalk network. The operator of one of the three systems is the team leader. The distribution of target attribute information among the team is configurable. In our experiments, each of the three operators has access to each target's speed, range, course, and bearing. Only the team leader has direct access to the target's IFF. Similarly, one subordinate has direct access to each target's altitude, whereas the other subordinate has direct access to each target's radar signal.

This dispersion of cue information makes it necessary for team members to exchange data, so that each operator has a complete cue set. For example, a team leader may have speed, range, course, bearing, and IFF for each aircraft. However, to make an optimal judgment, the leader must receive a radar signal and altitude from the subordinates. Similarly, for the subordinates to make optimal recommendations, they each need to receive IFF from their leader.

Information is exchanged via the message facility shown in Figure 3. Incoming messages are displayed in a scrollable window. The time of receipt, originator, track number of target for which the message is relevant, and type of message are displayed. The type of message can be a data message, a request for data, or a recommendation (received by the team leader only). A team member sends data to the other team members by entering the data in the appropriate box and clicking "SEND." The team leader makes the final decision on a target by selecting the threat level (1–7) and then clicking "SEND." Recom-

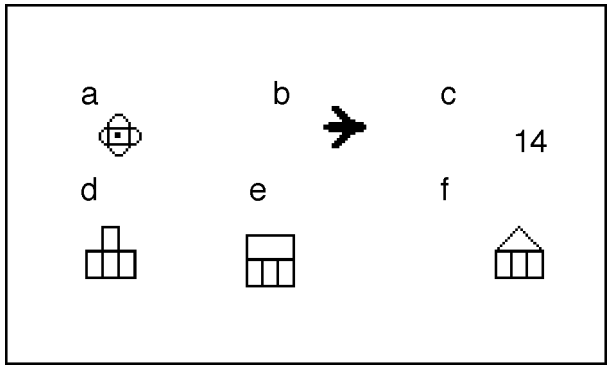


Figure 4. Icon displays available in Argus. Four sets of symbols have been used in Argus. Figure 4a is an example symbol from a set of standard symbols often used by the Department of Defense. Figure 4b represents a fighter from a pictorial set of symbols. Figure 4c shows that targets are sometimes represented by their track number. Figures 4d–4f are icons used in Team Argus. The different shapes of these icons represent different distances from a sector boundary.

mendations are sent from the team members to the team leader in the same manner.

Even though extensive training is given on the scoring algorithm, in Argus Prime we provide a help facility. Clicking on the attribute label in the information window pops up a window containing the mapping of raw to cue values for that attribute.

In both systems, 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. In Argus Prime, immediate feedback can be given each time a target is classified. As described in the Experiments section, the form of this feedback has varied with each Argus Prime experiment.

For both systems, data saved to the log file include all subject actions and all information necessary to reconstruct what was on the display at the time of each interaction. In addition, for Argus Prime, eye-tracking data and mouse position data are sampled and saved 60 times per second. For both systems, the data set is sufficient to "replay" a movie of all simulation events and subject actions. For Argus Prime, this movie includes eye movements and mouse movements.

EXPERIMENTER INTERFACE

Argus provides extensive support to the experimenter. A menu-driven interface gives the experimenter access to a wide variety of tools for scenario development, building sets of targets, changing how the game is scored, distributing work load among team members, and providing automatic counterbalancing of the experimental design.

Scenarios

Scenarios are files that specify the events that occur during the experimental session. Event-driven scenarios give the experimenter control over the number of targets on the screen at any given time. Scenarios may be of any length; to date, we have used multiple short scenarios (12 to 15 min) over multiple sessions (two to seven sessions). This section describes the events that can be scripted in a scenario and the development tools available to the experimenter to create and maintain scenarios.

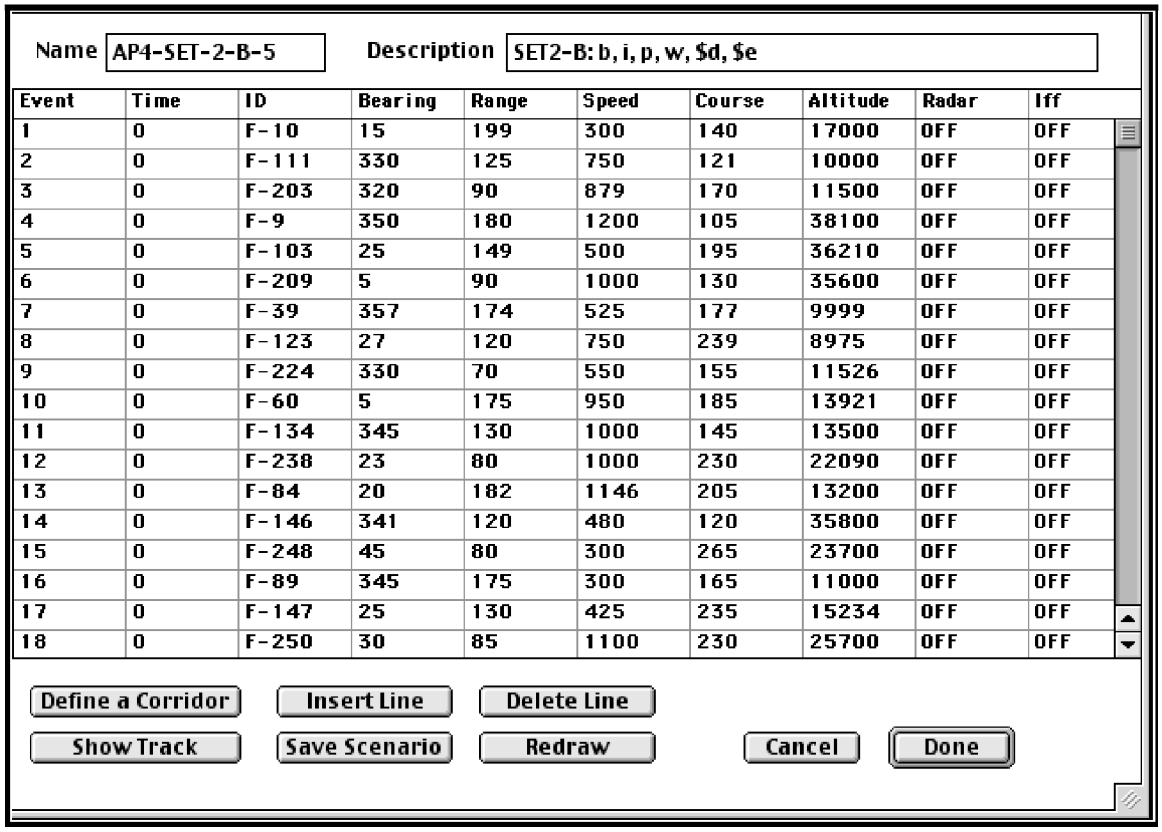
An event is an instruction to the software to change the state of the simulation, interface, or system. The scenario controls the dynamic aspect of the simulated radar display. When a target is added to the simulation its time of appearance and its dynamic attributes (range, bearing, speed, altitude, course, radar, and IFF status) are specified. During scenario execution, the range and bearing of the target are updated automatically by the system on the basis of the target's speed and course. All attributes, other than range and bearing, can be changed by a scenario event. The change can occur at a specific time or take place when the target crosses into the next segment.

Certain interface conditions can be controlled though scenario events. A system parameter can be set or cleared in the scenario (e.g., the change-color parameter, which, when set, causes the color of a target to change when the target is classified). The target attributes that are displayed in the information window can be controlled by the hide/show scenario event.

In Team Argus, all team communication is through messages. A message can be a data message in which target data available to one member are sent to the other team members. Likewise, other team members may send a message requesting a particular item of target data. Finally, subordinates can send the team leader recommendations for what classification to assign a target. When the system receives a message, an entry is added to the message queue in the information window (see the upper-right of Figure 3). The message can be read only after the team member who receives it selects the message from the input queue. Since all communication is through messages, a team member can be simulated by a scenario by scripting the messages that a real team member might send. (Indeed, this facility was used in the third Team Argus study.) The facility to script a team member allows precise control over the communication strategies used by the simulated team member.

In addition to creating scenarios, the interface allows the experimenter to edit and test scenarios. Scenarios can be built using the create/edit scenario tool shown in Figure 5. This tool supports iterative scenario design by providing a plotting tool that allows the experimenter to plot on the radar display the trajectory that a target will follow.

Scenario building is an incremental process since it is difficult to imagine how the screen will look when filled with targets. To assist in this process, Argus has the capability to test the scenario by running it at 2 to 10 times normal speed. (At 10 times normal speed, a 10-min scenario would take 1 min to run.) This allows the experimenter to ensure that the targets have sufficient separation and that they appear at the expected time and in the expected location on the screen.



Event	Time	ID	Bearing	Range	Speed	Course	Altitude	Radar	Iff
1	0	F-10	15	199	300	140	17000	OFF	OFF
2	0	F-111	330	125	750	121	10000	OFF	OFF
3	0	F-203	320	90	879	170	11500	OFF	OFF
4	0	F-9	350	180	1200	105	38100	OFF	OFF
5	0	F-103	25	149	500	195	36210	OFF	OFF
6	0	F-209	5	90	1000	130	35600	OFF	OFF
7	0	F-39	357	174	525	177	9999	OFF	OFF
8	0	F-123	27	120	750	239	8975	OFF	OFF
9	0	F-224	330	70	550	155	11526	OFF	OFF
10	0	F-60	5	175	950	185	13921	OFF	OFF
11	0	F-134	345	130	1000	145	13500	OFF	OFF
12	0	F-238	23	80	1000	230	22090	OFF	OFF
13	0	F-84	20	182	1146	205	13200	OFF	OFF
14	0	F-146	341	120	480	120	35800	OFF	OFF
15	0	F-248	45	80	300	265	23700	OFF	OFF
16	0	F-89	345	175	300	165	11000	OFF	OFF
17	0	F-147	25	130	425	235	15234	OFF	OFF
18	0	F-250	30	85	1100	230	25700	OFF	OFF

Figure 5. Create and Edit Scenario Screen. Events are listed in sequence based on the time at which they occur. All events shown represent initial target appearances. The Insert Line, Delete Line, and Redraw buttons at the bottom are editing aids. The Show Track button brings up the radar screen and draws the projected path of the selected target(s). The Save Scenario button writes the scenario to a file.

Targets

Targets do not have to be created anew for each experiment or scenario. Rather, targets may be stored in a library, so that targets with a certain combination of properties can be withdrawn as needed. For example, Team Argus has a target library of over 300 different helicopters, fighters, and bombers, all of which are classified by their hostility. Argus Prime has its own database with over 700 targets. The experimenter interface provides functionality to add targets, delete targets, and view the database. A target record in the database contains the static information about a target. Each target is given an identification code that can be used in a scenario to index the target. A target can be designated as having friendly, hostile, or unknown intent. The type of target can be specified as fighter, bomber, helicopter, or surface-to-air missile.

Scoring

The threat value assigned to a given set of target attributes can vary between experiments. For instance, in one experiment, a target's threat value may be determined solely by the target's distance from ownship. In another experiment, distance from ownship may be simply one among several attributes that are equally weighed to determine hostility. In this way, Argus enables the world to be as simple or complex as the experimenter wishes.

This flexibility is achieved by varying the scoring algorithm. The scoring algorithm consists of two parts: calculate a total score for a target and then map this score onto the 7-point threat scale. The total score is a weighted combination of cue values. Cue values are computed for target attributes. An example mapping for three attributes

Table 1
Example Mappings of Raw Values to Cue Values

Attribute	Raw value	Cue Value
Range		
	0–49	3
	50–99	2
	100–149	1
	>149	0
Altitude		
	0–15000	3
	15001–25000	2
	25001–35000	1
	>35000	0
Approach Distance		
	0–49	2
	50–99	1
	100–300 or 9999	0

is shown in Table 1. A cue value represents the degree of threat on a scale of 0–3, with 0 being the least threatening. Each cue value represents a range of raw values. As Table 1 shows, a cue value of 0 for range might be calculated for all range values greater than 149 miles, 1 for raw values of 100–149 miles, 2 for values 50–99 miles, and 3 for values 0–59 miles. This mapping of raw values to cue values is under the control of the experimenter.

The cue value for an attribute is multiplied by the weight that the experimenter gives the attribute to form one term of the total target score. A term may also be a weight times a product of cue values. For example, the cue value for range, 2, and the cue value for altitude, 3, may be multiplied together (yielding 6) and weighed twice so that the contribution of these factors to the total score for that target is 12. The total target score is derived by summing all terms in the algorithm.

After it is computed, the total target score is mapped to a threat value on a 1–7 scale. The mapping of this total target score to threat value is also under experimenter control. An example mapping is shown in Table 2. As an example of the scoring algorithm, suppose that the experimenter has determined that the target score has three terms: one for range, one for altitude, and one for approach distance, with weights 2, 1, and 2, respectively. At some point in the scenario, a target's range is 120 miles, its altitude is 17,000 feet, and its approach distance is 45 miles. Using Table 1, we calculate that the range cue value is 1, the altitude cue value is 1, and the approach distance cue value is 2. Therefore,

$$\text{target score} = (2 * 1) + (1 * 1) + (2 * 2) = 7.$$

The threat value is then determined from Table 2 to be 4.

Team Configuration

In Team Argus, the experimenter can specify the configuration of each team. One team member is designated as the team leader, and the others are team members. The distribution of target attributes that each team member sees is part of the design of the experiment. Typically, each team member will not have direct access to two of the attributes and must rely on each of the other team members to send data for that attribute. For example, one team member may need IFF data from the team leader and altitude data from the other team member.

Experimental Design

Argus facilitates the experimental design process by automatically assigning subjects to conditions in both within-subjects and between-subjects designs. Argus also automatically counterbalances conditions and scenarios.

Table 2
Mapping Target Score to Threat Value and Corresponding Action

Target Score	Threat Value	Action
0–1	1	Ignore
2–3	2	Review
4–5	3	Monitor
6–7	4	Warn
8–9	5	Ready
10–11	6	Lock-on
12–13	7	Kill

DATA OUTPUT AND ANALYSIS

Argus produces a log file containing all of the information necessary for data analysis and scenario debugging. Generally, one log file is created for each scenario session. For both Argus Prime and Team Argus, the log files are input to a preliminary data analysis program that collapses the data by subject or by scenario and produces an Excel file. Data analysis programs have also been developed to playback the log file.

Log File

Each log file has three parts: a parameter section, a scenario section, and an event section. The parameter section contains the labels and values for system parameters and contains model parameters if the system user is a simulated agent (as in the third Team Argus study). The scenario section contains a listing of the scenario that was executed during this session. The event section contains one record per event. Each record begins with the event type followed by the system time. The remainder of the record is specific to the event type.

The event types in Argus are subject events, system events, target events, and eye data events. System events signify system actions, such as the display of a window. Subject events signify subject actions, such as mouse clicks.

Target events signal changes to target attributes, such as course or altitude changes. Since targets are continuously moving, every range or bearing change is not logged. However, target position is logged periodically. This period is settable with the default being 5 sec. A periodic summary of all the target data is also logged. These periodic summaries aid in the playback of the log file.

Subject event lines include the target being hooked and the screen object for which the hook occurred. (A subject clicks on a target to hook it for further processing.) To enter the threat value of the hooked target, the subject pushes a radio button. The classification is finalized by the subject pressing the enter button. The classification is canceled by the subject pressing the cancel button. Help is accessed by clicking on the attribute label. When feedback processing is available, the subject action to access the feedback is logged.

In Argus Prime, eye data and cursor position data are collected 60 times a second. Each record contains the status of the eyetracker, the pupil diameter, the eye-gaze x-coordinate, the eye-gaze y-coordinate, and the cursor position in screen coordinates.

Data Analysis Programs

Two types of data analysis programs have been developed for Argus Prime and Team Argus. The first type takes as input a folder of log files to be analyzed and produces a file of performance and reaction time output measures by subject, scenario, or event. The second type inputs a

log file and plays it back graphically including eye movements and mouse movements.

For Argus Prime, our research has benefited from four levels of data analysis. At the highest level, we derive summary measures for each subject over all scenarios. These measures enable us to characterize broad strategy differences for measures, such as total number of targets classified or reclassified, mean time to process a target, mean number of clicks (hooks) of already classified targets, number correctly classified, and so on. At the next level of data aggregation, we can look at the same measures by condition (for within-subjects designs) or by scenario (for between-subjects or within-subjects designs). At the unit task level, we can derive measures of the processes involved in hooking and classifying individual targets. For example, we can look at the relationship between the target just selected and the last target selected (e.g., are they neighbors or were they both just about to cross over into a new sector?), the time spent in classification, and so on. Below this level, we can measure point of gaze, scan patterns, reaction time, and so on.

In Team Argus, two types of performance and reaction time files are created. Individual subject files contain data for one team member. These files contain variables with one value per target per sector. The variables include cue variables, variables for target truth, and subject's judgment. Timing variables for target change of state, and subject information access are also contained in the individual data files.

Summary files are organized by scenario and team. These files contain output variables with one value per team member. These variables support performance analysis at the level of team informity. Team informity is the degree to which the team as a whole is apprised of all the relevant cue values associated with the dimensions on which the decisions are based (Hollenbeck et al., 1995; Hollenbeck et al., 1997).

The capability to play back a scenario log file allows the experimenter to analyze strategies at the macro and micro level. When a scenario is played back, the target positions, the eye-gaze positions, and cursor positions are displayed over the interface used in the scenario. Mouse clicks are also shown during the playback. At the macro level, an important function of scenario playback is to ascertain what strategy the subject is using to select a target for classification. At the micro level, scenario playback shows fine-grained interactions of eye and mouse movements with interface components.

Model Component

A computational cognitive model, which can act as an experiment participant, is incorporated into Argus Prime. The current model is an embodied model—that is, it has *hands* and *eyes*. Our model runs under ACT-R/PM (Byrne, 1999; Byrne & Anderson, 1998) with the Eye Movements and Movements of Attention Extension (EMMA; Salvucci, 2000). The ACT-R/PM architecture combines ACT-R's theory of cognition (Anderson & Lebière, 1998) with modal theories of visual attention (Anderson, Matessa, & Lebière, 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 Argus Prime model directly perceives and manipulates the interface that is used by human users. This feature cleanly separates the model from the interface (Ritter, Baxter, Jones, & Young, 2000) and can provide an effective means for interface designers to evaluate new designs or design changes.

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 model's interface to Argus Prime is the same as the human's interface.

EXPERIMENTS

To date, five experiments have been conducted using Argus Prime and three using Team Argus. In this section, we briefly describe these experiments to show the flexibility of the system as an experimental framework.

Argus Prime Experiments

In the first Argus Prime experiment, the task interface was manipulated. A 2 x 2 between-subjects study was run using a tabular or a graphical interface versus requiring subjects to compute cue values from the raw values of target attributes or providing them with the pre-computed cue values.

The second Argus Prime study looked at transfer between low versus high work load scenarios. (Unlike in the first study, in the second and all subsequent studies subjects were given raw values and required to compute the cue values.)

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 one of the radio buttons in the information window. For example, if earlier in the same sector, the subject had classified the target's threat value as 6, that radio button would still be highlighted (see Figure 2) when the target was reselected. When the target was no longer selected (i.e., after it crossed a sector boundary), the radio button was no longer highlighted. 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-Dot* interface.

In the first two studies, subjects frequently reselected already-classified targets. Their pattern of behavior suggested that, rather than remember which targets were already classified, subjects simply clicked targets until they found one whose dot was not on. It was unclear to us

whether this memoryless strategy (Ballard, Hayhoe, & Pelz, 1995) was adopted by choice or whether, under the conditions of the study, human cognition is incapable of retrieving target status information. Argus Prime 3 addressed this issue by manipulating working memory load as a function of the design of the interface. To vary the ease of retrieving history information (i.e., "Have I classified this target already?") from the display, two new interfaces were introduced. The *noChange-noDot* interface is similar to the noChange-Dot in that the target's on-screen icon does not change when a classification is made. It differs from noChange-Dot in that the information window contains no record (i.e., the dot is not there) as to whether a target has already been classified. In contrast, for the *Change* 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 second and third Argus Prime studies, immediate classification feedback was provided in the upper left-hand part of the radar screen. However, it was unclear from those studies, how often subjects attended to that feedback. Argus Prime 4 was conducted to address that issue. The fourth study adopted a 2×2 within-subjects design. We used the noChange-Dot and noChange-noDot conditions to manipulate the availability of information as to whether a target was already classified. The second factor was feedback versus no feedback. In the no-feedback condition, there was no immediate feedback to subjects regarding the correctness of their classification. To obtain feedback in the feedback condition, subjects were required to move the cursor to and click on a feedback window. (In both conditions, cumulative feedback continued to be provided as a percentage score.)

Argus Prime 5 was a 5-h study that combined several novel features. First, after each scenario, subjects were required to rate the work load imposed by that scenario using the NASA-TLX (Hart & Staveland, 1988). The presentation and scoring of the NASA-TLX is an integral part of the Argus Prime interface that is controlled via the Experimenter Interface Module. Second, during the last 3 h of the study, subjects performed a secondary tracking task in the area to the right of the radar display and beneath the information window. Immediate feedback was provided during the scenario by changes in the tracking icon's color. Cumulative feedback (percentage of time that the airplane is being tracked) was provided at the center of the right-hand side of the screen. Summative feedback was provided at the end of each scenario. The Experimenter Interface Module controlled the sensitivity of the tracking task.

Team Experiment

For the first Team Argus study, we recruited 21 Army Reserve Officer Training Corps (R.O.T.C.) cadets for seven 2-h sessions. The cadets were placed in 3-person teams, with the highest ranking cadet serving as team leader.

Following 2 weeks of training, each team completed a series of scenarios over the following 5 weeks. The number of targets per minute, or tempo, was manipulated such that over the 5-week period, tempo was steadily increased. Group feedback was based on the target classification made by the team leader. In addition, the data we collected allowed us to assess the accuracy of the decisions made by each team member. As with Argus Prime, all events were collected, time-stamped, and saved to a log file. In addition, at the end of each weekly session, retrospective self-reports were collected to assess team perceptions of strategy development and adaptation.

The second Team Argus study used three interface conditions, Original Interface, Perceptual Aid, and Cognitive Aid, to collect data from three teams each over an 8-week period. (In addition, a slightly different set of cues and rules was used than those used in first Team Argus study.) The interfaces differed in the amount and type of information displayed to the subjects.

In the Perceptual Aid condition three aids were provided. First, an audio alert sounded when a new target appeared on the screen. Second, the color of the target had three states that indicated whether the target was untouched, hooked, or classified. Third, the shape of the target's icon also had three states. The shapes indicated the time left to classify the target (>30 sec, 15–30 sec, or <15 sec).

In the Cognitive Aid condition, feedback was provided to the subjects at the end of each scenario. For each team member, the overall performance, percentage of decisions made, accuracy of the decisions that were made, and how well each cue was utilized were displayed to the subject. In addition, the team leader received feedback on the relative agreement of each team member's recommendation, with the final decision made by the team leader.

In the third Team Argus study, the timing of information received from other team members was controlled by running the experiment with 1 human subject and 2 simulated team members. The simulated team members were the team leader and 1 subordinate. The timing of messages was controlled by using the Experimenter Interface Module (discussed earlier) to add message events to the scenario. In addition, the target icon changed appearance when a unique piece of information was sent to or received by the human member.

COMPARISON WITH OTHER SIMULATED TASK ENVIRONMENTS

Argus was designed to study how interface design affects cognitive work load. For the range of studies that we intended to conduct, we designed Argus with certain system features that would support the study of certain cognitive functions. In this section, we compare the cognitive functions and system features of Argus with those of other systems.

For cognitive functions, a key requirement for Argus was to enable us to study the interplay between cognition, perception, and action that takes place at about one third of a second—that is, at the embodiment level (Ballard, Hayhoe, Pook, & Rao, 1997). Whereas both Space Fortress (Donchin, 1995) and Advanced Cockpit (Ballas et al., 1992) enable investigators to focus on elementary cognition, perception, and action operations at this level of analysis, neither includes a judgment and decision-making task.

Second, at the embodiment level, we wished to study rapid shifts in serial attention. Both Space Fortress and Advanced Cockpit also support such shifts. To some degree, Tandem (R. P. Deshon, personal communication, February 20, 2001) requires shifts in serial attention; however, Tandem does not collect the data that would facilitate the detailed analysis of such shifts.

Argus' classification task incorporates judgment and decision making. Since we borrowed this task from TIDE^2 and Tandem, it should not be surprising that these systems also support the study of judgment and decision making.

A fourth major cognitive function that Argus was designed to study is cognitive work load. Space Fortress and Advanced Cockpit also facilitate the study of work load. Again, however, neither incorporates a judgment and decision-making task. Although Argus essentially borrowed the judgment and decision-making task used by TIDE^2 and Tandem, those systems do not enable the collection of process data, only outcome data.

Table 3 compares system features of the five systems. All except Space Fortress are scenario driven—that is, the experimenter can script scenarios for what events (with what set of attributes) occurs at what time. However, for both TIDE^2 and Tandem, once a target is created and set in motion, its attributes cannot change (S. M. Gully, personal communication, February 22, 2001). For example, once set in a certain direction at a certain speed, the targets must maintain that direction and speed until they leave the scenario.

Although three other systems enable some scripting of scenarios, only for Argus has there been considerable development of a set of direct-manipulation tools to facilitate the development, editing, and testing of scenarios.

Argus inherits parameterized scoring from TIDE^2 and Tandem. A target is not simply hostile or friendly; rather, it is classified on the basis of multiple attributes into one of seven categories in the friend-to-foe dimension. Likewise, a classification is not simply right or wrong, but, like the game of horseshoes, partial credit may be given for being close. Similar to TIDE^2 and Tandem, Argus can be used by individuals or teams.

Argus Prime incorporates an optional secondary task: tracking a moving airplane. This task is integral to Argus Prime in the sense of being built into the same interface as the classification task, providing cumulative feedback to subjects in the same manner, and saving performance data to the same log file. A secondary task is a required part of Advanced Cockpit. Although there is nothing to prevent the experimenter from imposing a secondary task on subjects in TIDE^2, Tandem (S. M. Gully, personal communication, February 22, 2001), or Space Fortress, such a task is not integral to these systems.

Table 3
Feature Comparison of Argus With Other Simulated Task Environment Systems

	System				
Feature	Argus	TIDE2	Tandem	Advanced Cockpit	Space Fortress
Scenario					
Driven	Yes	Yes	Yes	Yes	No
Create					
Edit/test	Extensive direct manipulation interface	Minimal DOS-based interface	Minimal DOS-based interface	Minimal nondisplay-based interface	Minimal DOS-based interface
Parameterized scoring	Yes	Yes	Yes	No	No
Modes	Single or team	Single or team	Single or team	Single	Single
Secondary Task	Optional	Not integral	Not integral	Always	Not integral
Data Collection					
Actions	Yes	No	Partial	Yes	Yes
Resolution	16.667 msec	n.a.	1 sec	16.667 msec	Unknown
Eye movements	Yes	No	No	No	No
Mouse movements	Yes	No	No	No	Joystick
NASA TLX	Yes	No	No	No	No
Playback	Yes	No	No	No	Yes
Multiple interfaces	Yes	Yes	Yes	Yes	No
Feedback					
Individual targets	Yes	No	Yes	Minimum	Yes
Summative	Yes	Yes	Yes	Not shown to subjects	Yes
Computational model	Yes	No	No	No	No

Note—n.a., not applicable.

From an experimenter's perspective, Argus stands out from the pack in its facilities for data collection. All user actions are collected and time-stamped with a resolution of 16.667 msec. Such actions can include eye movements and the position of the mouse. In addition, an option in Argus is the collection of estimates of subjective work load using the NASA-TLX (Hart & Staveland, 1988).

In common with Space Fortress, but distinct from all other systems, Argus supports the complete playback of scenarios with user actions. These actions include a representation of the users point of gaze and location of the cursor, as well as all system events.

For all systems, different interfaces have been developed over time. However, only Advanced Cockpit and Argus were specifically designed to support multiple interfaces. For Argus Prime and Team Argus, the set of interfaces developed to date is available via the Experimenter Interface Module.

As Table 3 shows, Argus can provide feedback on the accuracy of classifications for individual targets, as well as cumulative feedback throughout the scenario on summative performance. For Argus, cumulative feedback is provided for the classification task and for the target-tracking task.

Finally, only for Argus have computational cognitive models been developed that use the same interface as used by human subjects. This capability enables Argus to collect, save, and analyze the same data from computer subjects as it collects, saves, and analyzes from human subjects.

CONCLUSION

Studying complex behaviors requires computer systems that support all phases of the research program. This required support includes providing the experimenter with the capability to vary the experimental and task environment, building and running simulated task environments, and providing a wide range of data analysis tools. The Argus system provides such a platform for research in understanding individual and team behavior under varying degrees of cognitive and perceptual work load. Research groups interested in using Argus for noncommercial purposes should contact the authors by e-mail.

REFERENCES

Altmann, E. M., & Gray, W. D. (2000). Managing attention by preparing to forget. In *Human Factors and Ergonomics Society 44th Annual Meeting* (pp. 152-155). Santa Monica, CA: Human Factors and Ergonomics Society.

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

Anderson, J. R., Matessa, M., & Lebière, C. (1997). ACT-R: A theory of higher-level cognition and its relation to visual attention. *Human–Computer Interaction*, **12**, 439-462.

Ballard, D. H., Hayhoe, M. M., & Pelz, J. B. (1995). Memory representations in natural tasks. *Journal of Cognitive Neuroscience*, **7**, 66-80.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., & Rao, R. P. N. (1997). Deictic codes for the embodiment of cognition. *Behavioral & Brain Sciences*, **20**, 723-742.

Ballas, J. A., Heitmeyer, C. L., & Perez, M. A. (1992). Evaluating two aspects of direct manipulation in advanced cockpits. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems* (pp. 127-134). New York: ACM Press.

Byrne, M. D. (1999). *ACT-R/PM Documentation* [On-line]. Available: http://chil.rice.edu/ byrne/RPM/docs/ index.html

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

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

Dwyer, D. J., Hall, J. K., Volpe, C., & Cannon-Bowers, J. A. (1992). *A performance assessment task for examining tactical decision making under stress* (Special Rep. 92-002). Orlando, FL: Naval Training System Center.

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

Gray, W. D. (in press). Simulated task environments: The role of high-fidelity simulations, scaled worlds, synthetic environments, and micro-worlds in basic and applied cognitive research. *Cognitive Science Quarterly and Kognitionswissenschaft* (Special joint issue).

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (task load index): Results of empirical and theoretical research. In P. A. Hancock & N. Meshkati (Eds.), *Human mental workload* (pp. 139-183). Amsterdam: North-Holland.

Hollenbeck, J. R., Ilgen, D. R., Sego, D. J., Hedlund, J., Major, D. M., & Phillips, J. (1995). Mutlilevel theory of team decision making: Decision performance in teams incorporating distributed expertise. *Journal of Applied Psychology*, **80**, 292-316.

Hollenbeck, J. R., Sego, D. J., Ilgen, D. R., Major, D. A., Hedlund, J., & Phillips, J. (1997). Team decision making accuracy under difficult conditions: Construct validation of potential manipulations using the TIDE^2 simulation. In M. T. Brannick, E. Salas, & C. Prince (Eds.), *Team performance, assessment, and measurement: Theory, research, and applications* (pp. 111-136). Hillsdale, NJ: Erlbaum.

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**, 391-438.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer–Human Interaction*, **7**, 141-173.

Salvucci, D. D. (2000). A model of eye movements and visual attention. In N. Taatgen & J. Aasman (Eds.), *Third International Conference on Cognitive Modeling* (pp. 252-259). Veenendal, The Netherlands: Universal Press.

NOTES

1. Argus takes its name from the Greek god with a thousands eyes. Argus was given the task of guarding Juno after Zeus kidnapped her and tuned her into a cow. Juno was rescued by Mercury only after he succeeded in putting each of Argus' thousand eyes to sleep.

2. IFF is sent out by commercial aircraft and friendly forces to identify them as neutral or friendly.

(Manuscript received November 16, 2000;
revision accepted for publication March 12, 2001.)