Supporting Cognitive Models as Users

FRANK E. RITTER
Penn State University
GORDON D. BAXTER and GARY JONES
University of Nottingham
and
RICHARD M. YOUNG
University of Hertfordshire

Cognitive models are computer programs that simulate human performance of cognitive skills. They have been useful to HCI by predicting task times, by assisting users, and by acting as surrogate users. If cognitive models could interact with the same interfaces that users do, the models would be easier to develop and would be easier to apply as interface testers. This approach can be encapsulated as a cognitive model interface management system (CMIMS), which is analogous to and based on a user interface management system (UIMS). We present five case studies using three different UIMSes. These show how models can interact with interfaces using an interaction mechanism that is designed to apply to all interfaces generated within a UIMS. These interaction mechanisms start to support and constrain performance in the same ways that human performance is supported and constrained by interaction. Most existing UIMSes can and should be extended to create CMIMSes, and models can and should use CMIMSes to look at larger and more complex tasks. CMIMSes will help to further exploit the synergy between the disciplines of cognitive modeling and HCI by supporting cognitive models as users.

Categories and Subject Descriptors: D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing tools* (e.g., data generators, coverage testing); H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Evaluation/methodology*; *User interface management systems* (UIMS); I.2.0 [**Artificial Intelligence**]: General—*Cognitive simulation*; I.6.5 [**Simulation and Modeling**]: Model Development; I.6.7 [**Simulation and Modeling**]: Simulation Support Systems

General Terms: Design, Human Factors

Additional Key Words and Phrases: Cognitive modeling, usability engineering

Support has been provided by DERA and by the UK ESRC Centre for Research in Development, Instruction and Training. The views expressed in this article are those of the authors and should not be attributed to the UK Ministry of Defence.
Authors' addresses: F. E. Ritter, School of Information Sciences and Technology, Penn State University, 512 Rider Building, 120 S. Burrowes Street, University Park, PA 16801-3857; email: ritter@ist.psu.edu; G. D. Baxter, Department of Psychology, University of York, York Y01 5DD, U.K.; G. Jones, Institute of Behavioural Sciences, University of Derby, Mickleover, Derby DE3 5GX, U.K.; R. M. Young, Department of Psychology, University of Hertfordshire, Hatfield, Herts AL10 9AB, U.K.

1. THE SYNERGY BETWEEN COGNITIVE MODELING AND HUMAN-COMPUTER INTERACTION

Cognitive models—simulations of human behavior—now perform tasks ranging in complexity from simple mental arithmetic to controlling simulations of complex real-time computer-based systems, such as nuclear power plants. In the future these models will become even more powerful and useful.

Providing cognitive models[1] access to users' tasks can be done in three ways [Ritter and Major 1995]. In the first, the task simulation is implemented in the cognitive modeling language. This is the best approach when the focus of the modeling is mainly on the internal cognitive aspects and when the task is quite simple. Numerous models have used this approach (e.g., Beaman and Morton [1998], Newell and Simon [1972], Peck and John [1992], and Ritter et al. [1998]).

In the second method, a simulation of the user's task is implemented in a simulation language, but the interface or task cannot be directly used by human users, and the model's task simulation might not provide a visible representation of the task or of the model's behavior. The early versions of EPIC, for example, used this approach [Kieras and Meyer 1997], as do numerous other models (e.g., Ritter et al. [1998] and Taatgen [1998]).

In the third method, the task simulation can be operated by both the user and the model (e.g., Gray [2000]). This approach is superior when the simulation already exists, or when modeling complex dynamic domains where the task is too difficult to simulate using the previous approaches. It is preferable for the model to use the existing interface because fewer assumptions have to be made about equivalencies of the interfaces, and less work is required because only one interface is created.

We describe here an approach for allowing cognitive models more direct access to the interfaces users see. This is done by adapting ideas taken from user interface management systems (UIMSes) and extending them in a principled way so that cognitive models can interact with any interface built within the UIMS. This will allow cognitive models to be utilized as an interface design and evaluation tool on a wider range of interactive systems as previously envisioned [Olsen et al. 1993]. We next present the advantages of this approach for HCI and for modeling.

1.1 The Advantages for HCI

Cognitive models have been successfully used in three main ways by human-computer interaction (HCI) [John 1998]. The first way is to help examine the efficacy of different designs by using cognitive models to predict task performance times (e.g., Sears [1993]). The Model Human Processor, the Keystroke Level Model, and particularly the GOMS family of techniques [Card et al. 1983; John and Kieras 1996] have all been successfully deployed in the

[1]Unless specified, "model" refers to the model of the user, and "simulation" refers to the simulation of the task.

laboratory and in industry. These models can help create and choose better designs, sometimes saving millions of dollars (e.g., Gray et al. [1993]). The next step for this approach is to provide and include these and more complex models in design tools to provide feedback for designers.

The second way is by using cognitive models to provide assistance such as with an embedded assistant. In particular, models can be used to modify interaction to help users with their tasks. This technique has been employed in cognitive tutors (e.g., Anderson et al. [1995]). Some of these model-based tutors can be regarded as an example of what are commonly described as embedded agents or embedded intelligent assistants. The next step forward is to make the development of such models a more routine process.

The third way is by using models to substitute for users. These models are useful for populating synthetic environments [Pew and Mavor 1998], for example, to simulate fighter aircraft crews in a simulated war scenario [Jones et al. 1999]. In the future they will also lead to models that can test interfaces by behaving like a user. The next steps for this approach are to provide models with more realistic inputs and outputs mirroring human performance and to apply them more widely. Using models as users has been envisioned before (e.g., Byrne et al. [1994] and Lohse [1997]), but has not yet been widely applied.

One major impediment to each of these uses of cognitive modeling in HCI has been the difficulty of connecting the cognitive models to their task environment. Either the connection or the interface has to be built—sometimes both. There is an additional requirement that the connection should mimic the limitations and capabilities of human performance.

1.2 The Advantages for Models

Providing models with access to interfaces facilitates further development of the models and will open up new applications and expand existing ones. The main advantage is that the models will gain access to a much wider range of tasks than can be simulated in modeling languages. Early modeling work examined static tasks, keeping track of the task state in the model's head, at least partly because it is difficult to develop a model of an interactive task without providing the model with a capability for interaction with an external world (for a review of early models, see Ritter and Larkin [1994]). If they had been able to use the same interface as the corresponding subjects used, the models would have been applied and tested on more tasks and would have been able to cover a wider range of behavior, skills, and knowledge. These include models of computer interface usage [Altmann 1996; Bauer and John 1995; Howes and Young 1996; John et al. 1994; Peck and John 1992; Vera et al. 1993], as well as models of interaction with artifacts that could be simulated with a computer interface (e.g., air traffic control [Freed and Remington 1998], a VCR [Gray 2000]).

In addition to providing a richer world, creating a model that is embodied (i.e., with perception and motor actions) provides further constraint on a model. Closing the perceptual loop (or completing the perceptual cycle [Neisser 1976]), thereby restricting the model to interact only through its hand and eye, constrains the model's behavior to more closely resemble that of a real user [Baxter and Ritter 1997]. Although the constraints imposed by physical interaction may be relatively small on simple puzzle tasks, they are much more significant on interactive tasks. The models presented here must incorporate knowledge related to interaction such as where to look on the interface to find information. Because the models require a depth and range of knowledge about interaction, they predict that users do too.

Working with existing external simulations of tasks can make model development easier because it removes the need to create the task simulation using cognitive modeling languages, a difficult activity prone to error. Providing the model with access to the user's interface also leads to more accurate predictions. If the model and subject use the same interface there is less question about whether the model and the subject had access to the same material. There is also less development work required, because only one interface has to be created. Several of the case studies use simulations that already existed, thus relieving the modeler of the need to develop the task simulation from scratch. In addition, if the model can be used to suggest how to improve the interface, any changes can be applied to a single interface, rather than having to apply them both to the user interface and the model's representation of that user interface.

Finally, this approach will also lead to theory accumulation. In the examples we describe later, the models are developed using a cognitive architecture [Newell 1990], also referred to as an integrated architecture [Pew and Mavor 1998] when interaction is included. Cognitive architectures are theories of the common modules and mechanisms that support human cognition. They are typically realized as a programming language specifically designed for modeling, such as Soar [Newell 1990] or ACT-R [Anderson and Lebiere 1998]. Cognitive architectures offer a platform for developing cognitive models rapidly whilst still maintaining theoretical coherence between the models. Including interaction with an external world results in a more complete architecture.

Although there are tools to facilitate the development of user interfaces, and there are tools that can be deployed in the development of cognitive models, there are none that support connecting cognitive models to a wide range of interfaces. In the rest of this article we develop an approach to allow cognitive models access to the same user interfaces as users. Section 2 describes the cognitive modeling process, and introduces the concept of a cognitive model interface management system (CMIMS). Section 3 describes five example projects where models perform interactive tasks using the same type of simulated eye and hand implemented in three different interface tools and three modeling languages. These examples, when considered together with a review of related systems, suggest possible applications

Table I. The Artifacts Produced During the Cognitive Modeling of Interactive Tasks

Artifact	Purpose
Cognitive model	Simulates the cognitive performance and behavior of a human performing the task.
Task simulation	Provides the task, including the user interface that will be used by the cognitive model.
Linkage mechanism	Simulates human perception and action. Provides a way for the model and simulation to communicate.

and indicate where further work is necessary. Section 4 assesses the implications of the results of these projects, and identifies a number of ways in which integrated models could be exploited in the development of user interfaces, and more generally within HCI.

2. A ROUTE TO SUPPORTING MODELS AS USERS

The cognitive modeling process is unique in many respects, although the artifacts created by it bear some resemblance to products generated during the development of interactive software applications. We examine here the cognitive modeling process and introduce an approach to supporting cognitive models as users.

2.1 The Artifacts of the Cognitive Modeling Process

The cognitive modeling process, particularly as applied to the interactive tasks we are concerned with here, attempts to produce a cognitive model that performs like a human. The veracity of the cognitive model is tested by comparing its performance with human performance. The differences between the two are analyzed to understand why they occur, and then the cognitive model is appropriately refined, in an iterative cycle [Ritter and Larkin 1994].

The cognitive modeling process can be viewed as producing three artifacts, each of which fulfils a particular purpose, as shown in Table I. The first artifact is the cognitive model itself. As a theory, it has primacy.

The second artifact is a task application or its simulation. In simple, static task applications, such as small puzzles like the Tower of Hanoi, where the state of the task normally changes only in response to the user's actions, the task simulation can often be implemented using the cognitive modeling language. In dynamic tasks, however, where the state of the environment can evolve without outside intervention, the task simulation is best implemented separately. Where the original task is computer based, the simplest and most accurate approach is to allow the model to use the original task environment.

The third artifact is a mechanism that supports interaction between the cognitive model and the task simulation. The need for this linkage mechanism is most apparent in tasks in which the cognitive model has to interact with a task simulation implemented as a separate program, possibly running on a different computer.

There are existing tools that support the development of cognitive models, and of task applications. There are few tools that support the creation of the type of linkage mechanism required in cognitive modeling, however. User interface management systems are candidates to build upon.[2]

2.2 The Role of User Interface Management Systems for Supporting Cognitive Models as Users

To provide models with access to the same interfaces as users perhaps the best place to start is to consider tools used to develop user interfaces. In interactive software applications, the communication between the user interface and the underlying application is often implemented as a separate component. This component consists of a set of functions that provide a robust, uniform way of connecting the two. Cognitive models also require a set of capabilities that allow them to interact with the task simulation but that can be modified to approximate the same limitations and capabilities as humans have.

Any initial considerations for a tool kit to support the cognitive modeling process will therefore need to incorporate the following features:

—A tool to create interfaces.

—A run-time mechanism that lets the cognitive model interact with the task simulation (i.e., a model eye and hand).

—A communication mechanism that passes information between the cognitive model and the task simulation.

User Interface Management Systems (UIMSes) already provide a similar set of features for the development of interactive applications. UIMSes are systems designed to support interface creation and help manage the interaction when the interface is used (e.g., Myers [1995] and Open University [1990]). UIMSes can be used to create interfaces and applications in their implementation language, or can create interfaces that are tied to external applications. By definition UIMSes provide a set of features that very closely match our requirements.

UIMSes also offer a way to apply this work widely. They are designed to create multiple interfaces. Working within an UIMS will lead to the models being able to use any interface created with the UIMS.

2.3 Cognitive Model Interface Management Systems

The approach we are creating by extending a UIMS to support models as users can be described as a Cognitive Model Interface Management System (CMIMS), a system for managing the interactions of a cognitive model analogous to how a UIMS manages a user's interactions. The name CMIMS reflects the parallels with UIMSes, particularly the parallel needs between (human) users and cognitive models.

---

[2]An alternative basis for the linkage mechanism is to do image recognition of the screen directly, which some are now attempting to do [Zettlemoyer and St. Amant 1999].

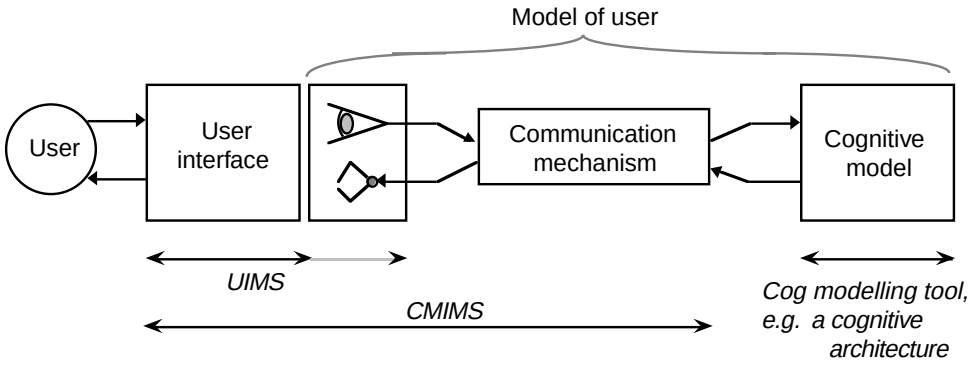

Fig. 1. Implementation of a cognitive model tied to a user interface of a task simulation, where the model and the simulation may be running in different environments (programming languages, processes, and/or computers). The hand and eye are implemented in the same UIMS as the task simulation, and communicate with the cognitive model via a communication mechanism.

Figure 1 depicts a CMIMS, showing the functional aspects of tying together a task simulation and a cognitive model. On the left of the figure is the user interface of a task simulation, and on the right is the cognitive model. The first step in getting the model to interact with the task simulation is to extend the cognitive model to be a more complete model of the user, by adding a simulated eye and a simulated hand to provide the model with capabilities for perception and action. We have found that the simulated eye and hand are best implemented in the same environment as the task simulation. The simulated eye needs access to the visible task objects (i.e., to the objects in the display) to create descriptions for the cognitive model, and the simulated hand needs to be able to implement the model's actions in the environment. The interface development tools within UIMSes provide facilities that support these functional capabilities. In particular, in UIMSes there are tools to find which objects occlude other objects (such as the simulated eye being over a textual label on the display), for the representation of objects such as mouse cursors, the capability to send mouse and keyboard actions to the interface, and the ability to create displays and control panels.

The second step is to link the cognitive model to the simulation, so that the model's eye can observe the simulation and pass back information to the cognitive model, and so that the model's hand can pass actions to the simulation. The end result is a model of a user in contact with a task environment, where information about the environment, and actions on the environment, is conveyed and constrained by the simulated eye and hand.

The resulting configuration is shown in the linked boxes across the middle of Figure 1. The model of the user is now split into two parts, with the simulated eye and hand implemented in the same software environment as the simulation, whilst the cognitive model is separate. Interaction between the model and simulated eye and hand occurs via a communication mechanism; its nature will vary depending on implementation language

and machine choice. Incorporating the simulated eye and hand into the UIMS potentially allows them to interact with any interface in the UIMS. Thus, it provides models with routine access to interfaces.

The arrows underneath the boxes represent the scope of the various tools. Particularly where the user's task involves interaction with a computer interface, the task simulation is well supported by standard UIMSes. The dashed extension to the UIMS arrow denotes the observation we made above, that the facilities needed to implement the eye and hand can usually be based on existing facilities in the UIMS. However, we will see later in the article that the simulated eye and hand place requirements that not all UIMSes or related systems currently satisfy.

Next, the CMIMS arrow reflects our suggestion for the development of Cognitive Modeling Interface Management Systems. As can be seen, in addition to the usual UIMS role, a CMIMS would need to include the hand and eye, as well as the communication mechanism between them and the cognitive model. Finally, on the right of the diagram is the cognitive model.

How cognitive architectures use the simulated eyes and hands is architecture dependent. Architectures will differ in how they represent and use the results of vision, and how they prepare to perform motor output. For all of the architectures, (1) visual search is required to find information to examine, (2) the amount of information available at any point is limited, and (3) performing visual search takes time. Similar restrictions apply to the motor output.

In the far left of Figure 1, the circle labeled *User* indicates that the cognitive model can work with the same task interface as users. This feature supports gathering data to test the model. It also indicates that the user can work with the model serving as a helper or agent within the user's interface.

Keeping the task simulation distinct from the cognitive model has three advantages. First, it makes development easier because the model and the simulation can then be tested and debugged independently. Second, it makes it less likely that the modeler will unintentionally incorporate assumptions about the task into the cognitive model, or about cognition into the task simulation. Third, it makes it easier to use different cognitive models with the same task simulation, or to apply the same cognitive model to different tasks. When a model performs a task in its own "mind," it is difficult to utilize the model or the task elsewhere, because they are specific to a detailed set of circumstances. Where a model is developed that works with one interface, there may be other interfaces to which it can be applied as well.

2.4 A Functional Model Eye and Hand

The Sim-eye and Sim-hand[3] are the most important part of the CMIMS. They bring the model into contact with the interface. We explain here the functional capabilities our Sim-eye and Sim-hand provide, and some empirical

[3]In order to emphasize the distinction between the model's capabilities and human capabilities, we will refer to the model's implementation of visual perception as the Sim-eye, and the model's implementation of motor action as the Sim-hand.

regularities they can easily support. Table II lists an initial set of functional capabilities and empirical regularities on which to base models of interaction. The table also indicates how many of these capabilities and regularities are demonstrated by the models described in Section 3, showing that with each additional refinement to the implementation (from left to right), the models had more capabilities and have exhibited more of the regularities.

The capabilities and regularities in Table II were chosen based on a literature review [Baxter and Ritter 1996] as the most important for an initial model of interaction. These are the capabilities and regularities that will have the most impact on cognitive models, either by providing them with new capabilities or by providing them with important and gross performance limitations, particularly for behavior over one second. These capabilities are fundamental necessities to support interaction.

Models exhibiting a greater number of empirical regularities can be created with these capabilities. This could, for example, include relative recognition rates of different colors. Other modelers may choose to focus on other regularities, such as those influencing behavior below half a second. The most important point is first to support the process of providing cognitive models access to interfaces in UIMSes by providing these functional capabilities—matching further empirical regularities can come later as refinements.

Providing functional capabilities first and then modifying the Sim-eye and Sim-hand to match more empirical regularities has proved to be a useful development strategy. For example, by giving the Sim-eye and Sim-hand visible representations, the modeler can observe their behavior on the display screen, and use these observations to refine the implementations of the Sim-eye and Sim-hand themselves.

Some of the other systems discussed in Section 3.6 are designed to look at dual-tasks, allowing cognition and interaction to occur in parallel. Often, they have not put development effort into their own usability and have not used more general UIMSes (such as SLGMS, Tcl/Tk, or Visual Basic). With time, these approaches will converge because they only represent different development priorities—none of the developers would argue, we believe, that usability or that accuracy are unimportant.

2.5 Sketchy Design

The Sim-eye and Sim-hand are controlled by the model through a simple command language, shown in Table III. The Sim-eye can be moved around the display using the saccade command to move the eye, and can inspect what appears at the current location on the display using the fixate command. The saccade and fixate commands are implemented using functions in the UIMS for handling mouse actions, manipulating display objects, and drawing. Once implemented, the Sim-eye can see (access) every object on interfaces built within that UIMS provided that the UIMS uses a regular representation of objects that can be accessed at run time. Users

Table II. Summary Functional Capabilities (CAP) and Empirical Regularities (REG) that Can Be Supported (adapted from Baxter and Ritter [1996]) and the Examples that Include Them

	Simulated Perception	ATC	Tabletop	Tower of Nott	CG/ Phone	EW Task	ACT-R/ PM*
Cap	A fovea represented and displayed.	Y	--	Y	Y	Y	Y**
Cap	A list of items sent to cognition upon fixation.	Y	--	Y	Y	Y	Y
Reg	A parafovea (5° to each side of the fovea) providing less information than the fovea.	--	--	Y	--	Y	N
Reg	A periphery reporting object location, moving or not, and a unique identifier.	Y	--	Y	--	Y	Y
Cap	Eye movement based on relative screen coordinates.	Y	--	--	Y	Y	Y
Reg	Eye movements take appropriate amounts of time.	--	--	Y	--	Y	Y
Cap	The eye can smoothly track an object.	Y	--	--	--	Y	Y
	Simulated Motor Action						
Cap	Mouse represented and displayed.	--	Y	Y	Y	Y	Y
Cap	Mouse button event actions: press and hold, release, click, double-click.	--	Y	--	--	Y	Y
Reg	Maximum speed for hand movements (e.g., 30 cm/s).	--	Y	Y	Y	Y	Y
Reg	Finite mouse button operation speed.	--	Y	--	Y	Y	Y
Reg	Adjustable typing speed.	--	Y	--	Y	Y	Y
Reg	Other typing regularities.	--	--	--	--	--	--
Reg	Time to move the hand between mouse and keyboard.	--	Y	--	Y	Y	Y
Cap	Continuous tracking, sequential, and repetitive movements.	--	--	--	--	Y	Y
Reg	Mouse move times based on two phase positioning movements. (Movements may or may not be modeled.)	--	Y	--	--	Y	Y
	Eye-Hand Coordination						
Cap	Response time of the eye and hand fast enough to control motor movements via visual feedback.	--	--	Y	--	Y	Y
Cap	Control panels.	--	--	--	Y	Y	--
Reg	Motor movements are sensitive to target size.	--	Y	--	--	Y	Y

*Except where noted, ACT-R/PM as presented by Byrne and Anderson [1998].

**The display became available in ACT-R/PM v. 1.0, released 99.07.08, www.ruf.rice.edu/~byrne/RPM/index.html.

Table III. Summary of Commands to Control the Sim-Eye and Sim-Hand

Sim-Eye Commands	Sim-Hand Commands
saccade <deltaX> <deltaY>	press mouse button <buttonName>
	release mouse button <buttonName>
fixate	click mouse button <buttonName>
	double click mouse button <buttonName>
	start mouse move <deltaX> <deltaY>
	update mouse move <deltaX> <deltaY>
	stop mouse move
	move hand from mouse to keyboard
	move hand from keyboard to mouse
	type character <character>

and the model can see the same display (to the limit of the theory of vision implemented in the Sim-eye).

Sim-hand also has a set of commands that allow the model to move its mouse around the display, and to perform mouse actions: press-mouse-button, release-mouse-button, and so on. The Sim-hand implementation will vary based on the UIMS' relationship to the operating system on which the Sim-hand is to be used.

In our models, which do not look at very rapid interaction, cognition generates an interaction command and then waits for the perceptual and motor operations to complete. While our models have used these capabilities in a synchronous way, this design allows cognition and interaction to occur in parallel.

We next examine how this functional model of interaction can support models as users. These five case studies show that it is possible to create CMIMSes, and that there are advantages in so doing.

3. EXAMPLE COGNITIVE MODELS THAT INTERACT

We have created a series of cognitive models that interact with task simulations. The five simulations were developed using tools that can be described as UIMSes. A different set of examples would yield a different set of lessons, but we believe only slightly different. We see many commonalties across this diverse set. We also review some other systems that model interaction.

3.1 Simplified Air Traffic Control Model

The first task simulation is a simplified air traffic control (ATC) task [Bass et al. 1995]. It was designed to explore how to create a general eye, and to let us understand what a model would do with an eye. We wanted a model in this domain to generalize to more complex tasks, and to assist us in understanding what knowledge should be included in models that interact. Such a model could help support the user by predicting what they would do, and then to assist them or do it for them.

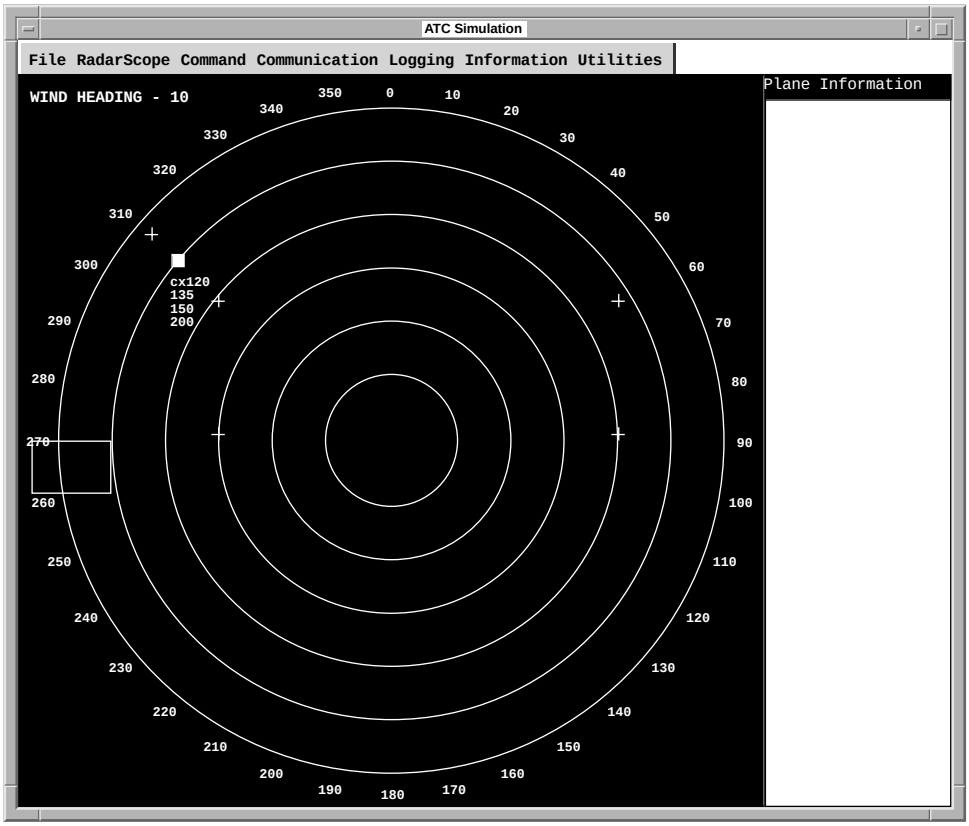

Fig. 2. The display of the ATC simulation showing the model's fovea (the white rectangle on the left-hand side of the figure), before it moves to the plane labeled "cx120".

We had access to ATC task simulators, but not to one that we could have our model interact with, let alone interact in a psychologically plausible way. A task simulation had to be developed, therefore, to allow the task to be performed both by the cognitive model and by users. The user interface, shown in Figure 2, is a simplified version of an air traffic controller's display screen. It includes some of the standard features that would appear on a real controller's screen, such as range rings. The current position of the aircraft is indicated by a track symbol (a solid white square) that has an associated data block depicting the aircraft identifier (cx120), its heading (135°), its speed (150 knots), and its altitude in hundreds of feet (e.g., 200 represents 20,000 feet).

The simulation provides a simplified version of an approach air traffic control up to the point where aircraft would normally be handed over to ground controllers. So, for example, when an aircraft is directed to change its heading, the turning time is not based on a detailed model of aircraft behavior. The task simulation does, however, provide facilities that allow the model to control the behavior of the aircraft by instructing it to change its speed, heading, and altitude. When an aircraft comes close enough to

the airport it is removed from the display and tagged as having successfully landed.

The basic task involves learning how to direct a single aircraft to land at an airport located at the center of the display. The choice of flight path is based on reading the wind speed and direction. The aircraft has to be guided along a path identified by a number of way markers, which appear on the screen as crosses. In order to land the plane, the cognitive model directs the aircraft to change its heading as it approaches each of the way markers on its path. A crucial element of the task is that the change of heading commands must be issued at the appropriate time, which requires that the cognitive model be able to detect when an aircraft is approaching a way marker.

The simulation was implemented using the Garnet UIMS [Myers et al. 1990]. Garnet was chosen because it was familiar and provides fairly general support for creating interfaces. The model was implemented in Soar [Laird et al. 1987].

3.1.1 *Visual Perception and Action*. The Sim-eye was implemented as part of the user interface developed within Garnet. The visible representation of its data structure consists of a transparent rectangle outlined in white (representing the area of the screen that would normally project onto the fovea of a real eye, the area of most acute vision, which is about the size of a thumbnail when viewed at arm's length). When a fixate command is sent by the model, the details of the objects appearing inside the foveal rectangle are sent back to the cognitive model as symbolic descriptions. This Sim-eye includes a coarse level of vision outside the fovea, providing to cognition only the location of objects that appear outside the fovea.

The Sim-eye is moved around the ATC display window (shown center left in Figure 2) by the model placing saccade commands to be processed into the Soar-IO facility. When the Sim-eye saccades, the list of visual elements is removed from cognition. The Sim-hand was implemented here as simple function calls to the task simulation via the Soar-IO facility.

3.1.2 *The Communication Mechanism*. The communication mechanism is implemented by a system called MONGSU [Ong 1994], which is based on Unix sockets. MONGSU allows any pair of Lisp or C-based processes to communicate using list structures and attribute-value pairs. Here, it was the ATC simulation in Lisp and the model in Soar (a C-based process).

3.1.3 *Summary*. The ATC model demonstrated that a simple but functional Sim-eye could be created using an existing UIMS. Knowledge about when and where to look at the screen is domain dependent. For this reason, knowledge about the ATC display had to be included in the cognitive model. Moving the Sim-eye around to see objects on the display slowed down the performance of the model because it had to work to get information—all the problem information was not resident in the model. So, an apparently trivial task, such as finding the single number representing wind speed

from the screen, became an intricate process, involving its own level of problem solving and search based on the external interface.

The symbolic descriptions returned upon a fixation are based on the graphical object hierarchy in Garnet. It is easy to create a general Sim-eye when all the objects to be recognized are part of a hierarchy. Garnet's object hierarchy ensures that all objects can be recognized in the same way.

The use of sockets as a communication mechanism added an extra layer of complexity because it requires the modeler to start two processes instead of one. Establishing a connection that is external in some way is also more prone to human and network errors than working within a single process.

This model used knowledge that has not often been seen in a cognitive model: where to look and how to visually monitor. The model had to find both the wind heading and the plane using its peripheral vision. Surprising to us at the time, but quite clear in hindsight, is that the eye is not just the fovea: the periphery is needed even for simple search, and otherwise the model has tunnel vision and must carefully scan the entire screen with its fovea. In a more complex display there will be many more objects. Knowledge must be available about where to look, even if it takes some searching to find objects. The cognitive model included new behaviors such as monitoring a plane, and could later reflect on and learn from its interactions. These behaviors suggest that knowledge acquisition studies of expert performance should not just examine what experts do but must also examine what they look for, and where and when they look.

3.2 Simple Tabletop Model

A simple task simulation was developed in Garnet as a test bed for developing a Sim-hand to provide models with a motor action capability [Rassouli 1995]. This tabletop task simulation is shown in Figure 3. It is implemented as a window with a number of different objects and a (model's) mouse to move between them. A simple Soar model was developed to demonstrate that a general Sim-hand created in a UIMS could be controlled by a cognitive model.

3.2.1 *Motor Action*. The Sim-hand has its own mouse pointer icon that moves around the tabletop based on commands issued by the cognitive model. The Sim-hand accepts requests to perform mouse operations by passing events to the window system. In this way the cognitive model can interact through typing and mouse actions.

3.2.2 *The Communication Mechanism*. The cognitive model is connected to the task simulation using the same interprocess communication mechanism (MONGSU) as had been previously used in the ATC task.

3.2.3 *Summary*. The Sim-hand demonstrated that a simple but functional hand could be created using an existing UIMS. The Sim-hand generated mouse button events (click, press, and release) in software mimicking the hardware mouse.

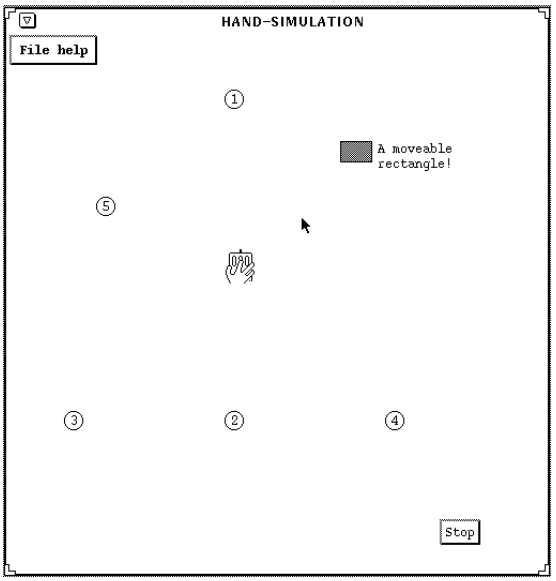

Fig. 3. The simple tabletop written in Garnet (from Rassouli [1995]). The model's mouse pointer is the hand with the mouse; the hardware mouse used by the modeler is the arrow.

Having to explicitly move the hand and click the mouse button makes the model perform the task interactively, bringing the model closer to human behavior by forcing the interaction behavior to be explicit rather than implicit. Clicking on a button is not automatic or a single step: the model had to work out where to move the hand, move it, and then click the mouse button.

The initial implementation of the Sim-hand could not select items from the menus it brought up. Garnet, like many graphics systems, requires interface objects, such as the Sim-hand's mouse pointer, to be attached to a specific window, and Garnet implements pull-down menus as separate windows. The initial Sim-hand mouse pointer could not be positioned over the menu items, which were in a different window. This problem can be fixed by putting a model's mouse pointer on every window or by moving the model's mouse pointer to the top-most window after every interaction or window change. This mouse/window problem illustrates how implementation details of the interaction model will be dependent on the system used (e.g., the lack of transparent windows in the X window system), as well as some possibly general solutions (e.g., keeping a software mouse pointer on every window).

When this project was complete, the Sim-hand was included in the ATC task model described above. The revised version of the ATC cognitive model helped to illustrate two of the difficulties involved in modeling task interaction: (1) some eye-hand coordination knowledge is needed, and further work needs to be done to explore the best way to gather this from subjects and include it in models; and (2) there is an opportunity to gather and include additional regularities about visual attention, including those

relating to mouse movements, such as moving the mouse to a flashing light or object.

3.3 Tower of Nottingham Models

The Tower of Nottingham is a puzzle where 21 wooden blocks are assembled to form a pyramid structure. The pyramid is constructed from five different size layers each comprising four blocks of the same size, with one pinnacle block. It has been used extensively to study how children's abilities to learn and communicate develop between the ages of 3 and 9 years (e.g., Wood and Middleton [1975]).

The model's and subjects' visual search in service of problem solving and object manipulations are analogous to manipulation in graphical user interfaces and to some aspects of graphic design in drawing and CAD/CAM packages. The extensive interactive nature of this model and its task, including learning while searching, is a fairly useful, simple way to explore some of the important and common issues in interaction in many screen-based manipulation tasks.

The blocks simulation is written in Garnet. The complexity of the task precludes simulating it with a cognitive modeling language. An initial model in Lisp was used to develop and exercise an initial simpler task simulation [Ritter et al. 1994]. The current model is written using the ACT-R (v. 3.0) cognitive architecture [Anderson and Lebiere 1998] and interacts with a completely enhanced version of the task simulation.

3.3.1 *Visual Perception and Motor Action*. The task simulation, shown in Figure 4, includes a Sim-eye and a pair of Sim-hands, which are represented graphically, making task behavior readily visible. The Sim-eye and Sim-hand are controlled using the commands previously listed in Table III, particularly, moving the Sim-eye to a location and fixating upon that location, and moving the Sim-hands. The command language was extended to enable the Sim-hands to grasp and release blocks (which allows them to fit, stack, and disassemble blocks) and to rotate blocks. Although this level of representation is more abstract than mouse movements it represents the atomic cognitive actions in this task, allowing the model and the modeler to represent interaction in an appropriate way. It avoids the level of individual finger movements, which while interesting, do not appear to be an important aspect of this task.

When the Sim-eye is requested to fixate, it passes back to the model the relevant blocks and block features as moderated by area of the eye the blocks are in. The blocks information is represented in the model as declarative knowledge, and overwrites the results of previous fixations. Objects that appear in the periphery of the Sim-eye are represented by an identifier only. Overwriting is a plausible theory of visual input [Horowitz and Wolfe 1998]. The quality of this approach will have to be tested with further models, but in this case it helps the model fit our data.

There is a small difference between the level of information available from the Sim-eye's fovea and parafovea. The features and sizes of blocks in

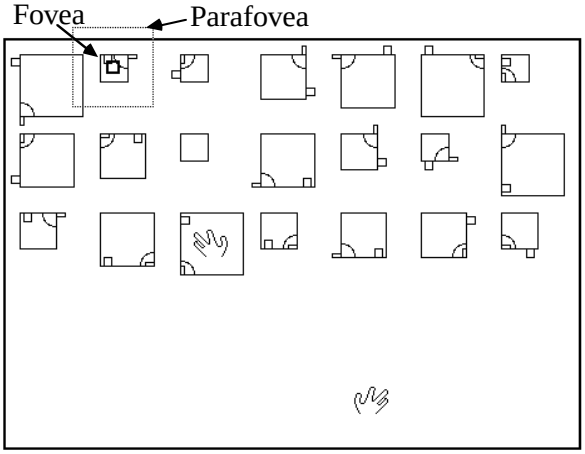

Fig. 4. The Tower of Nottingham. The fovea is the small black outlined square in the center of the block in the top left corner. The parafovea, shown as a dashed line, extends approximately two fovea widths in each direction beyond the fovea. The left Sim-hand has picked up one of the largest blocks.

the fovea are reported accurately, and those in the parafovea are subject to a small, adjustable amount of noise. This mechanism provides a way to mistake similar sizes and similar features. These mistakes are in accordance with adult performance on the task [Jones and Ritter 1998; Jones et al. 2000].

The model sends requests for the Sim-hands to perform actions, such as moving to a location. If a Sim-hand is grasping a block, then that block is also moved. Once an action is completed the cognitive model is informed. The model can then manipulate the Sim-eye to verify that the chosen action has been completed. Although actions are currently always completed successfully, it is especially useful for the Sim-eye to fixate when fitting blocks together or disassembling block structures, because this can change the model's representation of the current state of the task.

3.3.2 *The Communication Mechanism*. Interaction between the cognitive model and the task simulation is easily achieved using Lisp function calls because both are based on Common Lisp. The cognitive model sets up a goal in ACT-R for each interaction it wishes to perform. Each goal causes the appropriate function for the Sim-eye or the selected Sim-hand to be called. When the interaction is complete, the relevant goal in the cognitive model is flagged as having been achieved. The model can then terminate the goal and continue its behavior.

3.3.3 *Summary*. This case study further indicates that the designs of the Sim-hand and Sim-eye are generally applicable, and that they can be used by a different cognitive architecture, in this case, ACT-R.

Using a common language for the model and the task simulation makes them easier to implement, test, and run. The simplicity in interaction, the

fact that parts of the task simulation already existed, and the fact that an initial Sim-eye and Sim-hand had already been successfully used meant that Garnet was a very viable tool.

The importance of perception in task performance that had been found in the ATC model was confirmed. Explicitly controlling the Sim-eye and Sim-hands changed the model's behavior. Including perception and explicit motor actions forced the model to expend time and effort to find and assemble blocks. The model spent approximately half of its time interacting with the simulation, suggesting that any model for a task involving interaction requires an external task in order to accurately reflect human behavior.[4]

The Tower of Nottingham example demonstrates that it is possible and useful to model interactive tasks. The performance of the model matches the performance of adult subjects on the task reasonably well because the whole task was modeled and because the necessary learning could occur in visual search, in cognition, and in output [Jones and Ritter 1998].

We were able to examine more closely how problem solving develops. Several important developmental theories were implemented in the model, and their predictions compared with children's data, showing that differences in strategy choice is the most likely candidate for what leads to children's longer solution times in this task [Jones et al. 2000].

The model also predicts that some mistakes are caught before they are executed. The features of blocks in the parafovea are not always correctly seen. If a block is incorrectly seen as having the target features when it is in the parafovea, the eye will saccade to the block to prepare to pick it up, and the hand will move to the block as well. When the block image is located in the fovea, the correct features will be seen, and the action will be abandoned. This behavior of moving hands to blocks but not picking them up seems to occur in adult data, and suggests there are types of mistakes that are not fully overt. This type of mistake could occur for adults using interfaces as well.

Modeling the complete task allowed us to compare the results directly with quantitative regularities, including learning across layers (for further details see Jones [1999] and Jones et al. [2000]). As an example, Figure 5 shows the mean time taken to construct each layer for adult subjects and the model. There is a high correlation between the performance of adults and the model for the mean time taken to construct each layer ($r^2 = 0.92$). Comparisons are also favorable for the RMS error for each layer (which indicates the average percentage difference between the model scores and subject scores for each layer). The RMS error is low for both the time to construct each layer (4.1%) and the number of construction attempts made in producing each layer (5.7%). This comparison would have

[4]This is detailed in a submitted paper by Jones, G. and Ritter, F. E., "Over-estimating cognition time: The benefits of using a task simulation."

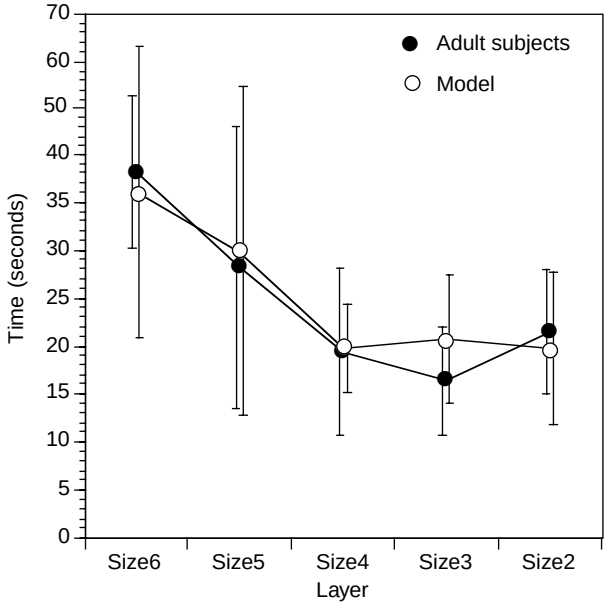

Fig. 5. Time taken (in seconds) by adults and the model to complete each layer. Error bars are to the left for adults, to the right for the model.

been impossible or less reliable without basing it on the model with a Sim-eye and Sim-hand.

3.4 Tcl/Tk Models

Tcl is a widely used programming language with Tk an accompanying set of graphical extensions for creating graphical interfaces [Ousterhout 1994] and user interface design environments such as SpecTcl. There are extensive examples available in the numerous textbooks and Web sites devoted to it. The latest full release of Soar (v. 7) incorporates Tcl/Tk. This makes it possible to develop a task simulation that could be directly and easily linked to a Soar model.

Two models have been created developing a CMIMS for Soar based on Tcl/Tk. The first model created was one of exploratory search. When moderately experienced Apple Macintosh users are faced with using an application they have not previously encountered, they carry out an exploratory search to find objects and associated actions that can help them achieve the task at hand [Rieman et al. 1996]. A model of this exploratory search had been initially developed in Soar (v. 6), with the simulation of the task—generating a chart using the CricketGraph program—also being written in Soar. When the same model (in Soar 7) was tied to a simulated version of CricketGraph its limited behavior was more apparent.

This system also included for the first time control panels to drive the eye and hand independently of the cognitive model. The control panel also displayed the current state of the Sim-eye and Sim-hand, making debug-

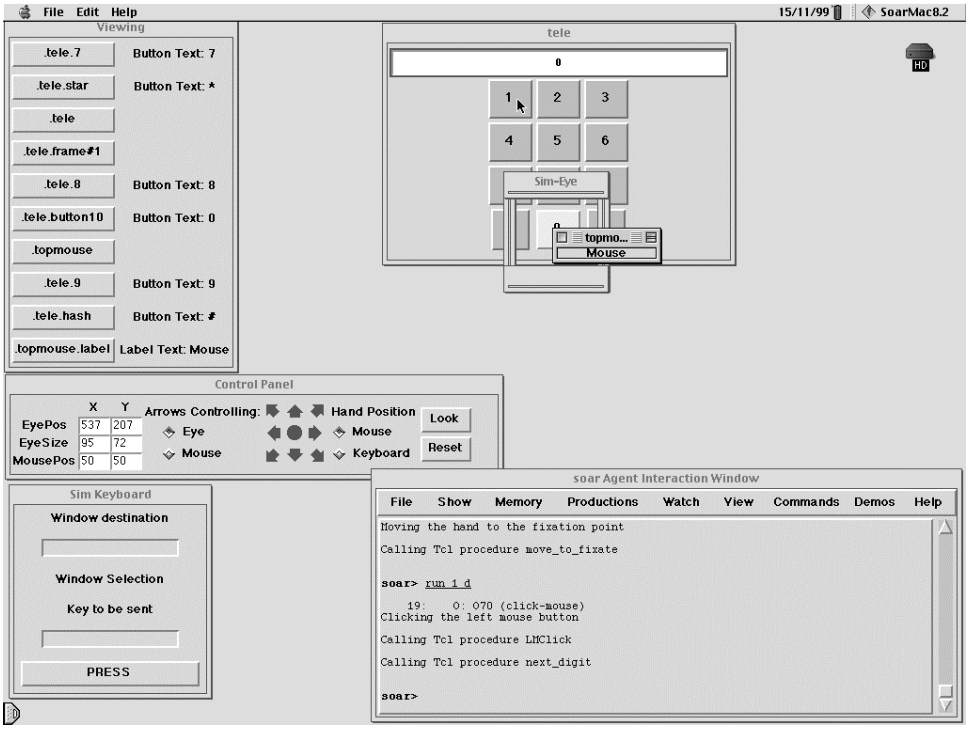

Fig. 6. An example phone interface (top center), the CMIMS control panels (left), and the Tcl/Tk Soar Interface (bottom right). The fovea is focused on the "0" button on the phone, although more objects are visible in the fovea as indicated by the list displayed in the window labeled "Viewing." The model's mouse pointer's hot spot is over the "0" button. The modeler's mouse pointer is over the "1" button.

ging easier because the state was visible and because the Sim-eye and Sim-hand could be manipulated prior to programming the model.

The second model implements a simple model similar to GOMS [Kieras 1997] to dial numbers from memory on five graphical phones [Harris 1999; Lonsdale 1999]. A sample phone and what the modeler sees are shown in Figure 6. The model is designed to be able to dial any number on any phone written in Tcl/Tk. This will break down, of course, in interfaces that are novel, but starts to represent in a procedural and runnable way what knowledge is necessary, and to make the knowledge reusable across multiple interfaces. The model shows improvement with practice, and predicts that some misdialing errors are due to moving the mouse into the center of the fovea when the target button is on the edge of the fovea. The model's predictions of time to dial a standard UK number of 11 digits (ranging from 15.4 seconds to 30.2 seconds depending on practice and relative fovea size) are similar to NGOMs' [Kieras 1997], but are higher than actual times (approximately 10 seconds). Like GOMS, though, the relative times between interfaces should allow useful comparisons.

3.4.1 *Visual Perception and Motor Action*. A Sim-eye reports information from the interface. A square box outline indicates where the model is looking. The Sim-eye is visibly represented as a rectangle made up of four line-shaped windows joined together because Tcl/Tk does not easily support overlaying objects. These problems are similar to the problems that arose with the tabletop model's Sim-hand in that the solution is not completely straightforward, but in these general environments a solution was available. While not all the areas of the eye are modeled, the groundwork to implement them has been created.

3.4.2 *The Communication Mechanism*. Interaction between the cognitive model and the task simulation is easily achieved using Tcl function calls because they are in the same language. The cognitive model puts its command on the Soar-IO link, and it is passed each model cycle to the Sim-eye or Sim-hand. Results are put back on the Soar-IO link for use by cognition.

3.4.3 *Summary*. The Sim-eye and Sim-hand here demonstrate that a simple but functional hand could be created in a widely used UIMS. The ability to interact with Tcl/Tk interfaces opens up a wider world of applications to Soar models because of the range of existing and potential interfaces in Tcl/Tk. The cognitive model of exploratory search is now being expanded to develop a more complete model of search in real interfaces (i.e., those not written by the model developers). While these models are simple, they point the way to models that could be used to interactively test interfaces.

A regular object hierarchy is what allows the Sim-eye to be general. The Tk data structures are not as regular as those in Garnet.[5] This lack of uiformity requires a more elaborate algorithm to find what the Sim-eye should see.

The control panels greatly help with understanding what the cognitive model has to achieve and with debugging. They make it very clear why the phone model sometimes misdials. The panels also make it possible to test the Sim-eye and Sim-hand separately. The panels allow visual search strategies to be tried manually before implementing them in the model.

3.5 Electronic Warfare Task Model

The final example is work in progress, in which we have attempted to build on the lessons learned from our earlier models. The Electronic Warfare (EW) task is to protect a ship from enemy missiles by maneuvering the ship, jamming the radar of incoming missiles, and firing chaff to decoy the missiles away from the ship [Chapman et al. 1996].

The interface for the EW task is shown in Figure 7. The task simulation, OOPSDG [Ramsay 1995], was developed in Common Lisp. The user interface and the Sim-eye and Sim-hand were developed by Tenet Systems using

[5]A summary is available from http://www.psy.herts.ac.uk/pub/R.M.Young/.

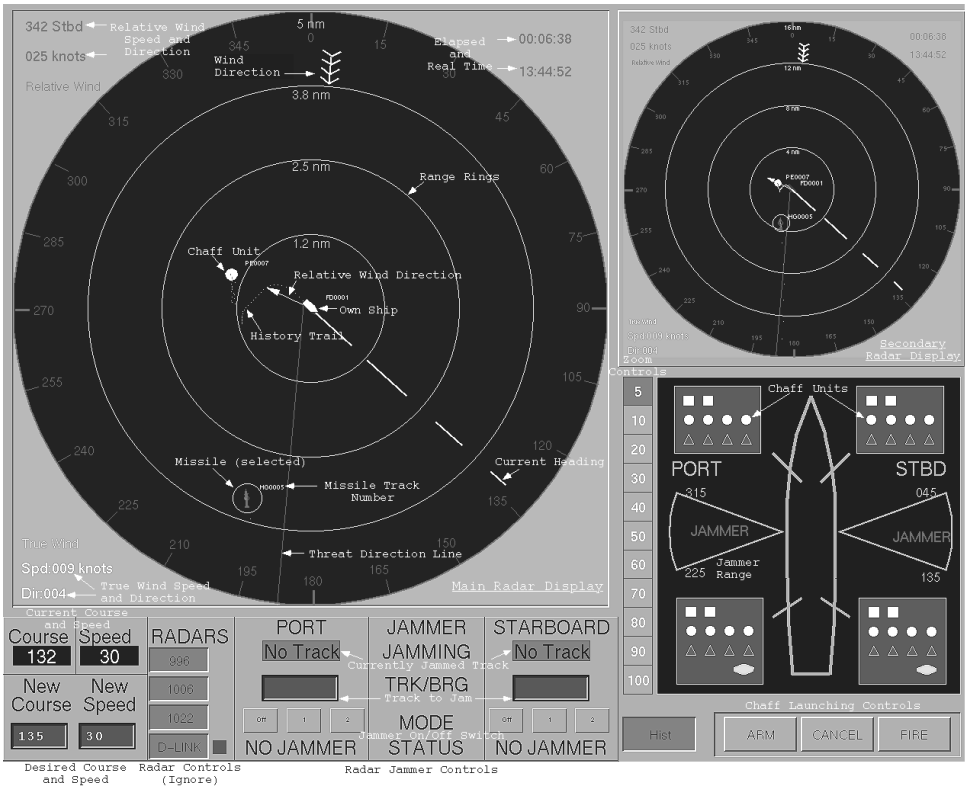

Fig. 7. The EW task interface, annotated with descriptions of the various components.

Sherrill-Lubinski's object-oriented, real-time dynamic Graphical Management System, SL-GMS [Sherrill-Lubinski Corporation 1994]. Figure 8 shows a simplified interface that was used to develop the Sim-eye and Sim-hand in the upper window and the control panel in the bottom window.

There are several reasons for creating a cognitive model of this particular task. It could help examine the efficacy of different designs. Also, we would have a model of what the user was trying to do. This would support exploring new ways to provide assistance, including manuals, tutoring systems, and embedded assistants.

When the model's behavior is compared with that of the user to validate the model, using the same interface ensures consistency. The EW task has a complex interface, and duplicating it would be difficult and error prone. If the model can be used to suggest how to improve the interface, working with the same interface means that changes do not have to be duplicated in two interfaces; it also allows the model to work with the user as an assistant.

3.5.1 *Visual Perception and Motor Action*. The Sim-eye has been extended to provide perception of objects located in parafoveal and peripheral vision with several configurable options to facilitate theory development,

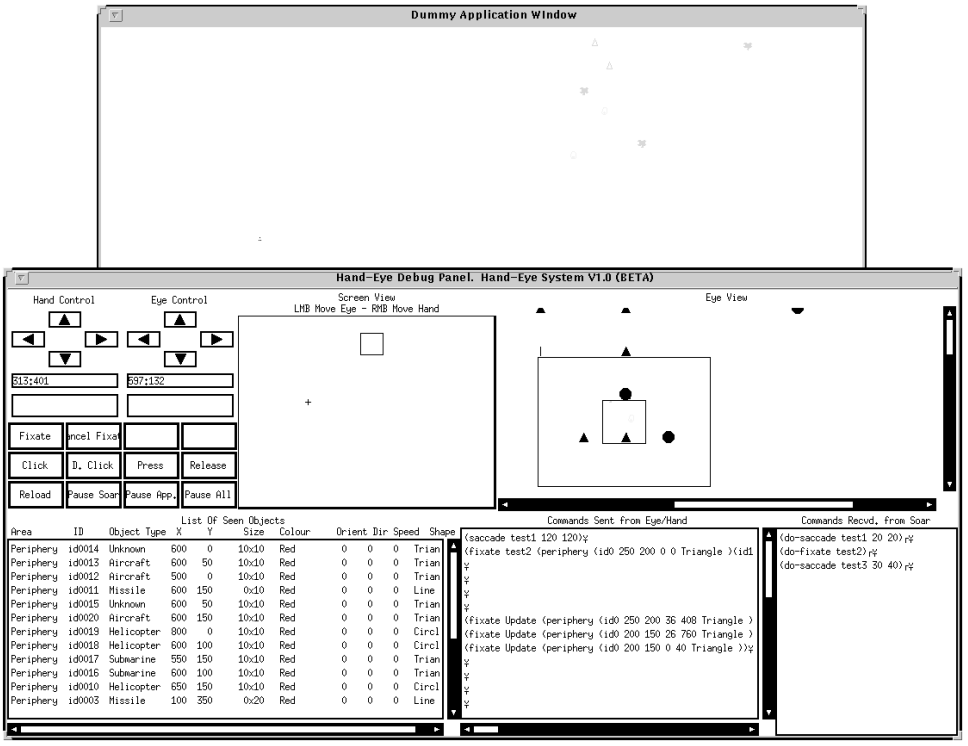

Fig. 8. The test display (top) for developing the eye and hand for the EW task in SL-GMS and the control panel (bottom). Normally these two windows are on different monitors. The control panel includes manipulation controls (upper left of this window), and continuing along its top, two displays showing the current position (screen view) and detailed contents of the fovea (eye view). Along the bottom are listings of the output of the eye to the model, and the commands sent through the linkage mechanism.

such as fovea size and how shape and color interact to attract attention. By default, the detail provided for objects appearing in the parafovea is less than that of objects appearing in the fovea, but greater than that of objects appearing in the periphery. Only a few features like location and motion are available for peripheral objects, whereas shape, color, orientation, size, and location are available for foveal objects. There were many new types of objects in this task, so this implementation allowed for the eye to be appropriately adjusted based on further relevant experimental or published information. The extended version of the Sim-eye has as its default settings the Sim-Eye in the Tower model.

It is difficult in SL-GMS to create an extra mouse pointer for use by the model. Having a single mouse would be a problem if only one display screen is used because the cognitive model and the cognitive modeler would have to share the mouse. Using two display screens removes this conflict, and provides additional display space. One of the displays is dedicated to the running application with its full-screen display, including the Sim-eye and

Sim-hand, whilst the other display provides the control panel that allows the Sim-eye and Sim-hand to be monitored and manually controlled.

3.5.2 *The Communication Mechanism*. The connection between the cognitive model and the task simulation is based on the ideas in the MONGSU interprocess communication utility that we used in the ATC and tabletop examples. The cognitive model is being implemented in Soar (v. 7.1), which incorporates Tcl/Tk and hence provides built-in commands to manage socket communication. The communication between the model and the task simulation implements the command language described in Table III.

3.5.3 *Summary*. Even though this cognitive model is currently incomplete, we have already learned several lessons. First, the transfer of the general Sim-eye and Sim-hand design to another development environment provided further evidence of the generality of this design. The Sim-eye and Sim-hand were implemented in SL-GMS in just two weeks. Also, this demonstrated that our approach to interaction can be understood and implemented by UIMS programmers.

Using two separate display screens, with the simulation (driven by the cognitive model) on one screen and the control panel (driven by the modeler) on the other, solves several problems. During debugging there is a need for the modeler to be able to view the interaction control panel and the task simulation simultaneously. By using a dedicated display screen, the control panel can be extended to incorporate the additional debugging facilities necessitated by the increased complexity of the EW task interface. Using separate screens allows the cognitive model to control the Sim-hand's mouse pointer on one screen, whilst the modeler has control of the mouse pointer for the control panel on the other screen. Without this capability the modeler cannot query the model while it runs (with the mouse), and the model and modeler come into conflict trying to use the same mouse.

An initial model of this task was implemented in Soar before the Sim-eye and Sim-hand implementations became available. None of the model's 70 production rules scan the display or perform interaction. It is clear that tying this model to the task in Figure 8 will profoundly change the task from a geometry task of computing intercept angles to a scanning and monitoring task with geometry as only a subcomponent.

3.6 Related Systems

Other examples where models interact with simulations provide further lessons. These systems as well as our examples are summarized in Table IV. The columns correspond to objects in Figure 1. The related systems presented here have used three UIMSes, and typically can present displays on more than one type of machine. The simulation languages are usually closely tied to the UIMS, but in the EW task a simple simulation is available in SLGMS and a more complete one in OOPSDG. There are a variety of communication mechanisms, but direct function calls are the

Table IV. Summary of Examples

	UIMS	Display	Simulation Language	Linkage Mechanism	Cognitive Modeling Language
ATC	Garnet	X windows/Mac	Common Lisp	UNIX sockets	Soar
Table top	Garnet	X windows/Mac	Common Lisp	sockets	Soar
Tower of Nottingham	Garnet	X windows/Mac	Common Lisp	function calls	Lisp/ ACT-R
CG/Phone	Tk and/or SpecTcl	X windows/PC/ Mac	Tcl/Tk	function calls	Soar
EW Task	SLGMS	X windows	SLGMS and/or OOPSDG	sockets	Soar
Soar agents/ MIDAS/ APEX/ Driver-Soar	custom interfaces	varies	varies	varies	varies
EPIC	none	none*	EPIC interface simulation language	function calls	EPIC production system
ACT-R/PM	Macintosh Common Lisp	Mac	Common Lisp	function calls	ACT-R

*Later versions have a visual display.

easiest to use. Soar, Lisp, and ACT-R have been used to implement the models.

The Soar agent simulations [Jones et al. 1999], which support large-scale military training exercises, and MIDAS [Corker and Smith 1993], which is a design tool for aviation systems that includes a model user, are further illustations of how cognitive models can be useful. They interact directly with simulations without a perceptual/motor filter (via function calls for actions and data structures for perception). This approach to interaction has the advantages that it is easy to implement; it is not tied to a specific cognitive architecture; and, most importantly, it can quickly provide a rich environment and task for the models (which is the primary focus of this work). Interacting through simple function calls, however, fails to provide as much constraint on cognition. In this approach as well, humans cannot necessarily use the same systems, which reduces the ability to test and validate the models.

APEX [Freed and Remington 1998] is a design tool to predict what errors operators would make in complex domains. It has been applied so far to air traffic control systems and cockpits. It provides cognitive models in its own architecture, a communication mechanism, and simulated eyes and hands. It is not based on a UIMS, and users cannot interact with the same interface; modelers, however, can see the interface. APEX starts to model the effects of interaction, how visual displays can support problem solving, and how errors can arise in air traffic control.

Driver-Soar [Aasman and Michon 1992] is a detailed model of driving. It includes a model written in Soar that interacts with its own software car to

navigate through simulated road intersections. In addition to a simulated eye, Driver-Soar includes interaction modalities not addressed here, including head movements, further hand movements, and feet. These interaction modalities are implemented both in a Pascal-based and a Lisp-based simulation. While there are displays for modelers to watch the model's behavior, users cannot interact with the simulation in the same way. Driver-Soar's predictions have been compared with detailed human measurements, showing that such predictions can be quite accurate.

The models and simulations in APEX and Driver-Soar are not designed for reuse. They illustrate, however, some of the possible applications and results that are available from employing models as surrogate users.

EPIC [Kieras and Meyer 1997] is a system addressing the intricate details of perception and action. The cognitive models are written in its own production system that can communicate directly with a model eye and hand that interact with an interface simulator. Models do not access a visual display shared with users. Interfaces to be examined are implemented separately using a special production system to provide information to the model at either set times or upon set conditions. Users cannot interact with the interface the model sees. EPIC can be used to make accurate predictions of interaction behavior such as menu use [Kieras et al. 1997]. Some of EPIC's capabilities and the regularities they support have been used by Soar models [Chong and Laird 1997] and by ACT-R/PM [Byrne and Anderson 1998].

The only other system that could properly be described as a CMIMS is ACT-R/PM [Byrne 1999; Byrne and Anderson 1998]. It is a theory of perception and motor behavior realized in Macintosh Common Lisp. It provides an environment in which ACT-R models can interact with task simulations (psychological experiments, for example) that humans can use as well. The interaction is modeled on a detailed level, down to 50 ms—we have neither designed nor tested our functional models for this level of precision. The generality and utility of the ACT-R/PM analysis of perception and action have been demonstrated through multiple use by models [Byrne and Anderson 1998; 1999; Gray 1999].

ACT-R/PM is similar in many ways to the previous models of interaction we have built. We believe that part of the reason for the success of ACT-R/PM is that it provides the facilities required by a basic CMIMS. The ACT-R/PM model of perception is based on graphical objects in Macintosh Common Lisp so it can include a graphic display that can be seen by the modeler and used by subjects. Furthermore, its reuse—the incorporation and adaptation of some of EPIC's results, particularly the general structure (i.e., parallel execution of perception and action) and the motor component—are consistent with the approach of theory reuse that we advocate.

The major difference, if there is one, lies in the choice of priorities. ACT-R/PM is more concerned with detailed psychological predictions, but is not yet positioned to be a tool that can be widely used to develop user interfaces for two reasons: (a) ACT-R/PM was not designed to interact with every interface that can be built in Macintosh Common Lisp [Byrne and

Anderson 1998]; however, it can already recognize most objects, and it can be extended by the modeler; (b) ACT-R/PM is not in a major graphic interface tool. In the context of Figure 1, it provides a linkage mechanism but does not include a common or widely portable UIMS.

3.7 Limitations of this Approach

There are several limitations to the current generation of systems that could be classed as CMIMSes. The examples presented here cover only a small subset of all possible tasks and interfaces. As a functional model, these implementations of the Sim-eye and Sim-hand intentionally do not cover all that is known about interaction, nor do they include all forms of interaction. These models do not yet include fine-grained behavioral regularities or those that are based on emergent perceptual phenomena, for example, recognizing blank space as a region. When we have used these Sim-eyes and Sim-hands more, we will be in a better position to know where we need to extend the accuracy of perception and motor actions. In certain tasks, having a simpler representation of behavior will be useful (e.g., checking the function of an interface, qualitative learning effects) in the way that Newtonian mechanics is compared with quantum mechanics. Even the simulated eyes and hand in EPIC and ACT-R/PM are based on simplifying assumptions such as the use of a command language. It is almost certainly a simplification (albeit a useful one) to assume that perception, cognition, and action are this separate (e.g., see Pylyshyn [1999]).

The problem most often raised with respect to using models to test interfaces is that the interface must be completely specified before the model can be applied. There are several responses to this limitation. First, the limitation does not appear to be insuperable, but it would be an entirely separate project to apply models to sketchy designs (e.g., Szekely et al. [1993]). Second, there are many systems and approaches requiring a full design before their analysis can be done. Interfaces may be particularly prone to requiring a full specification before their use and efficiency can be estimated (e.g., Gray et al. [1998]). Third, this approach will put designers in touch with the limitations of users in testing preliminary designs. With experience, the designers may learn to avoid problems, based on their experience with the model user. Finally, tests of the interface are immediately informative, and problems can be directly rectified. An unusual advantage of this approach to testing interfaces is that, unlike electrical circuits, testing is done with the actual system.

4. COGNITIVE MODELS AND INTERFACES IN THE NEW MILLENNIUM

Supporting cognitive models as surrogate users is possible. The case studies have shown that the Sim-eyes and Sim-hands can be used by a variety of models interacting with a range of interface tools. It is now possible to apply theoretically grounded cognitive models to real-world HCI tasks.

Building a Sim-eye and Sim-hand for each computational cognitive model might be as ad hoc as building each new model in Lisp—you could lose the constraints imposed by an Integrated Cognitive Architecture. Here, eyes and hands have been built in several UIMSes from the same design. This reimplementation of the same approach provides a form of constraint because it is the same design that is being reused and because the capabilities and regularities are noted explicitly. The more important aspect is that the Sim-eye and Sim-hand are now available in several widely used software tools. This will allow them to be reused in a variety of interfaces and will provide much greater constraint and reuse. There are several CMIMSes in various UIMSes, so reuse should become a possibility for models in the future.

Modelers should use these Sim-eyes and Sim-hands in order to provide their models with an interactive capability. Newer cognitive architectures, such as Jack [Busetta et al. 1999], should attempt to provide at least one CMIMS for their models to use. ACT-R/PM's hand is available at http://www.ruf.rice.edu/~byrne/RPM/; the Tcl/Tk eye/hand will be available at http://ritter.ist.psu.edu.

UIMS designers should include support for cognitive models as users in their tools. The functional capabilities and experimental requirements in Table II show what is necessary to support cognitive models as a type of user and some of the experimental regularities that can be included. This list will help create models of interaction in other UIMSes.

We can now review this approach, noting how it can contribute to the development of cognitive models and what it means for the future of interfaces.

4.1 The Implications for Models

The models presented here would not have been possible to develop without access to external simulations. Having these models interact with interfaces has led to the ability to create models of far more complex tasks than if the tasks had to be simulated in the cognitive modeling language. Tasks too complicated to simulate in a cognitive modeling language are straightforward to simulate in a UIMS. A wider range of human behavior can be explored and modeled with this approach.

Having a Sim-eye and Sim-hand to interact with the outside world has lead to a greater sophistication and accuracy for these models. Including a theory of interaction has both provided models with more capabilities, and constrained the speed and abilities of the models, in a way approximating human behavior. CMIMSes provide a way to encapsulate these constraints.

Interacting has required the addition to the models of new knowledge and new types of knowledge, including where to look and what to do with it. This knowledge suggests that there are types of knowledge that the user has to know that is not often taught or referenced. When users do not know where to look, they have to search through the interface or use peripheral vision. The amount of knowledge and effort it typically takes the models to

interact suggest that the difficulty of what users have to do has been consistently underestimated.

The case studies also show that this approach supports several kinds of reuse. Multiple models can use the same interface (e.g., through the Sim-eye and Sim-hand with the Tower of Nottingham simulation). The same model can use multiple interfaces (e.g., the Soar phone model). Models, as well as users, can work with the same interface (e.g., the ATC task, the EW-Task). This approach will contribute to the reuse of models envisioned for cognitive architectures [Newell 1990] and thus be a further constraint on the architecture. This approach can be used to make more complete cognitive architectures.

There are several other systems that model interaction. The approach to modeling interactive tasks that we have adopted falls somewhere between the extremes of allowing models to directly access the internals of the task simulation and modeling interaction in full psychological detail. Focusing on functional capabilities has allowed us to apply this technique more widely, but the next step will be to incorporate more experimental regularities to model human performance more closely. Enforcing further experimental regularities (as summarized in ACT-R/PM and EPIC) on the functional capabilities we have created in Tcl/Tk would provide a system that both Soar and people could use, and one that has numerous existing interfaces and tasks.

4.2 The Implications for Interfaces

The potential benefit of deploying cognitive models as surrogate users during interface development is large. There are at least two significant ways in which CMIMSes can be exploited by cognitive modeling to facilitate the improvement of user interfaces.

First, cognitive models can be used to evaluate user interfaces. By using cognitive models in place of people, we could start to ask *what-if* questions about user interfaces, such as changing the interface and examining the effects on task performance. The models of phone interaction are starting to do this [Lonsdale 1999]. In this way, it becomes possible to dynamically evaluate *how* an interface is used, and where important events like errors may occur [Freed and Remington 1998]. Cognitive models of interface use, such as IDXL [Rieman et al. 1996] and the other models of problem solving and task performance we have described, could be developed further and applied. The models can also be used to inform the design of user interfaces by indicating which parts of the interface are used the most or are hard to learn.

Second, the ability to embed more accurate user models opens up a range of applications, such as more accurate intelligent assistants to help novices. With the interaction process developed, it will allow more time to be spent on creating the applications. These embedded assistants would encapsulate knowledge about a new range of possible behaviors, that is, interaction. This knowledge would then be used to determine what the user should do

next, and provide appropriate assistance to the user when requested or if the user selected an inappropriate course of action.

Although there is a synergy between the disciplines of HCI and cognitive modeling, it has not yet been fully exploited. Several results and techniques in HCI have been discovered using cognitive modeling [John 1998], but few of the lessons from HCI have been reapplied to increase the understanding and application of the models. We have highlighted one particular area where we believe UIMSes can be exploited to help in the development of cognitive models. It will take time to learn how to take advantage of all the benefits that will come through supporting cognitive models as users.

ACKNOWLEDGMENTS

We would like to thank Joe Mertz, Josef Nerb, Sarah Nichols, Gary Pelton, Paul Hall, and David Webb who have helped implement these examples. We would like to thank Jans Aasman, Erik Altmann, Paul Bennett, Michael Byrne, Wayne Gray, Steve Sawyer, and several anonymous reviewers for comments.

REFERENCES

AASMAN, J. AND MICHON, J. A. 1992. Multitasking in driving. In *Soar: A Cognitive Architecture in Perspective*, J. A. Michon and A. Akyürek, Eds. Kluwer Academic, Dordrecht, Netherlands.

ALTMANN, E. M. 1996. Episodic memory for external information. Ph. D. thesis, Tech. Rep. CMU-CS-96-167. Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

ANDERSON, J. R. AND LEBIERE, C. 1998. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

ANDERSON, J. R., CORBETT, A. T., KOEDINGER, K. R., AND PELLETIER, R. 1995. Cognitive tutors: Lessons learned. *J. Learn. Sci. 4*, 2, 167–207.

BASS, E. J., BAXTER, G. D., AND RITTER, F. E. 1995. Creating cognitive models to control simulations of complex systems. *AI Simul. Behav. Q. 93*, 18–25.

BAUER, M. I. AND JOHN, B. E. 1995. Modeling time-constrained learning in a highly interactive task. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '95, Denver, CO, May 7–11), I. R. Katz, R. Mack, L. Marks, M. B. Rosson, and J. Nielsen, Eds. ACM Press/Addison-Wesley Publ. Co., New York, NY, 19–26.

BAXTER, G. D. AND RITTER, F. E. 1996. Designing abstract visual perceptual and motor action capabilities for use by cognitive models. Tech. Rep. 36, ESRC CREDIT. Department of Psychology, University of Nottingham.

BAXTER, G. D. AND RITTER, F. E. 1997. Model-computer interaction: Implementing the action-preception loop for cognitive models. In *Proceedings of the 1st International Conference on Engineering Psychology and Cognitive Ergonomics, Vol. 2*, D. Harris, Ed. Ashgate Publishing Company, Brookfield, VT, 215–223.

BEAMAN, C. P. AND MORTON, J. 1998. Modelling memory-updating characteristics of 3- and 4-year olds. In *Proceedings of the 2nd European Conference on Cognitive Modelling*, Nottinghmam University Press, Nottingham, UK, 30–35.

BUSETTA, P., RÖNNQUIST, R., HODGSON, A., AND LUCAS, A. 1999. JACK intelligent agents—components for intelligent agents in JAVA. *AgentLink News Lett. 2* (Jan.). Available at http://www.agent-software.com/white-paper.pdf.

BYRNE, M. D. 1999. ACT-R perceptual-motor (ACT-R/PM) version 1.0b.5: A users manual. Psychology Department, Rice University, Houston, TX.

BRYNE, M. D. AND ANDERSON, J. R. 1998. Perception and action. In *The Atomic Components of Thought*, J. R. Anderson and C. Lebiere Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

BYRNE, M. D. AND ANDERSON, J. R. 1999. Serial modules in parallel: The psychological refractory period and perfect time-sharing. Submitted for publication.

BYRNE, M. D., WOOD, S. D., FOLEY, J. D., KIERAS, D. E., AND SUKAVIRIYA, P. N. 1994. Automating interface evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems: "Celebrating Interdependence"* (CHI '94, Boston, MA, Apr. 24–28), ACM Press, New York, NY, 232–237.

CARD, S. K., MORAN, T. P., AND NEWELL, A. 1983. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Inc., Hillsdale, NJ.

CHAPMAN, T., RITTER, F. E., AND BAUMANN, M. 1996. Electronic warfare task manual. Working Paper WP-R3BAIA005/013, Cognitive Modeling Unit. Department of Psychology, University of Nottingham.

CHONG, R. S. AND LAIRD, J. E. 1997. Identifying dual-task executive process knowledge using EPIC-Soar. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates Inc., Hillsdale, NJ, 107–112.

CORKER, K. M. AND SMITH, B. R. 1993. An architecture and model for cognitive engineering simulation simulation analysis: Application to advanced aviation automation. In *Proceedings of the AAIA Computing in Aerospace 9 Conference*, American Association for Information Architects.

FREED, M. AND REMINGTON, R. 1998. A conceptual framework for predicting error in complex human-machine environments. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, M. A. Gernsbacker and S. J. Derry, Eds. 356–361.

GRAY, W., Ed. 1999. *Proceedings of the ACT-R Workshop. Human Factors and Applied Cognition*. George Mason University, Fairfax, VA. Available at http://hfac.gmu.edu/actr99/.

GRAY, W. D. 2000. The nature of processing and errors in interactive behavior. *Cogn. Sci. 24*. To be published.

GRAY, W. D., JOHN, B. E., AND ATWOOD, M. E. 1993. Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world performance. *Human-Comput. Interact. 8*, 3, 237–309.

GRAY, W. D., SCHOELLES, M., AND FU, W.-T. 1998. When milliseconds matter: Implementing microstrategies in ACT-R/PM. In *Proceedings of the 5th ACT-R Workshop*, Carnegie Mellon University, Pittsburgh, PA.

HARRIS, B. 1999. PracTCL: An application for routinely tying cognitive models to interfaces to create interactive cognitive user models. Department of Psychology, University of Nottingham. Bachelor's of Science thesis.

HOROWITZ, T. S. AND WOLFE, J. M. 1998. Visual search has no memory. *Nature 357*, 575–577.

HOWES, A. AND YOUNG, R. M. 1996. Learning consistent, interactive, and meaningful task-action mappings: A computational modelYoun. *Cogn. Sci. 20*, 3, 301–356.

JOHN, B. E. 1998. Cognitive modeling in human-computer interaction. In *Proceedings of Graphics Interface '98*, 161–167.

JOHN, B. E. AND KIERAS, D. E. 1996. Using GOMS for user interface design and evaluation: which technique?. *ACM Trans. Comput. Hum. Interact. 3*, 4, 287–319.

JOHN, B. E., VERA, A. H., AND NEWELL, A. 1994. Towards real-time GOMS: A model of expert behavior in a highly interactive task. *Behav. Inf. Tech. 13*, 255–267.

JONES, G. 1999. Testing mechanisms of development within a computational framework. Ph.D. Dissertation. University of Nottingham, Nottingham, UK.

JONES, G. AND RITTER, F. E. 1998. Initial explorations of simulating cognitive and perceptual development by modifying architectures. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, M. A. Gernsbacker and S. J. Derry, Eds. 543–548.

JONES, G., RITTER, F. E., AND WOOD, D. J. 2000. Using a cognitive architecture to examine what develops. *Psychological Science 11*, 2, 1–8.

JONES, R. M., LAIRD, J. E., NIELSEN, P. E., COULTER, K. J., KENNY, P., AND KOSS, F. V. 1999. Automated intelligent pilots for combat flight simulation. *AI Mag. 20*, 1, 27–41.

KIERAS, D. E. 1997. A guide to GOMS model usability evaluation using NGOMSL. In *Handbook of Human-Computer Interaction*, M. G. Helander, T. K. Landauer, and V. Prabhu, Eds. Elsevier Science Publishers Ltd., Essex, UK, 391–438.

KIERAS, D. E. AND MEYER, D. E. 1997. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Comput. Interact. 12*, 391–438.

KIERAS, D. E., WOOD, S. D., AND MEYER, D. E. 1997. Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Trans. Comput. Hum. Interact. 4*, 3, 230–275.

LAIRD, J. E., NEWELL, A., AND ROSENBLOOM, P. S. 1987. SOAR: an architecture for general intelligence. *Artif. Intell. 33*, 1 (Sept. 1987), 1–64.

LOHSE, G. L. 1997. Models of graphical perception. In *Handbook of Human-Computer Interaction*, M. G. Helander, T. K. Landauer, and V. Prabhu, Eds. Elsevier Science Publishers Ltd., Essex, UK, 107–135.

LONSDALE, P. R. 1999. Extending PracTCL to provide a more functional eye and hand for the Soar cognitive modelling architecture. Department of Psychology, University of Nottingham. Master's of Science thesis.

MYERS, B. A. 1995. User interface software tools. *ACM Trans. Comput. Hum. Interact. 2*, 1 (Mar. 1995), 64–103.

MYERS, B. A., GIUSE, D. A., DANNENBERG, R. B., KOSBIE, D. S., PERVIN, E., MICKISH, A., ZANDEN, B. V., AND MARCHAL, P. 1990. Garnet: Comprehensive support for graphical, highly interactive user interfaces. *IEEE Computer 23*, 11 (Nov. 1990), 71–85.

NEISSER, U. 1976. *Cognition and Reality*. W. H. Freeman and Co., New York, NY.

NEWELL, A. 1990. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.

NEWELL, A. AND SIMON, H. A. 1972. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

OLSEN, D. R., JR., FOLEY, J., HUDSON, S., MILLER, J., AND MYERS, B. 1993. Research directions for user interface software tools. *Behav. Inf. Tech. 12*, 2, 80–97.

ONG, R. 1994. Mechanisms for routinely tying cognitive models to interactive simulations. Tech. Rep. 21. University of Nottingham, Nottingham, UK. Available as ftp://ftp.nottingham.ac.uk/pub/lpzfr/mongsu-2.1.tar.Z.

OPEN UNIVERSITY. 1990. *A Guide to Usability*. Open University Press, Milton Keynes, UK. In association with the Department for Trade and Industry.

OUSTERHOUT, J. K. 1994. *Tcl and the Tk Toolkit*. Addison-Wesley Professional Computing Series. Addison-Wesley Longman Publ. Co., Inc., Reading, MA.

PECK, V. A. AND JOHN, B. E. 1992. Browser-Soar: a computational model of a highly interactive task. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '92, Monterey, CA, May 3–7), P. Bauersfeld, J. Bennett, and G. Lynch, Eds. ACM Press, New York, NY, 165–172.

PEW, R. W. AND MAVOR, A. S., Eds. 1998. *Modeling Human and Organizations Behavior: Application to Military Simulations*. National Academy Press, Washington, DC. http://books.nap.edu/catalog/6173.html.

PYLYSHYN, Z. 1999. Is vision continuous with cognition? The case for cognitive impenetrability of visual perception. *Behav. Brain Sci. 22*, 3, 341–365.

RAMSAY, A. F. 1995. OOPSDG modelling environment for the centre for human sciences. Rep. DRA/CIS/(SS5)/1026/9/2. DRA Portsdown.

RASSOULI, J. 1995. Steps towards a process model of mouse-based interaction. Department of Psychology, University of Nottingham. Master's of Science thesis.

RIEMAN, J., YOUNG, R. M., AND HOWES, A. 1996. A dual-space model of iteratively deepening exploratory learning. *Int. J. Hum.-Comput. Stud. 44*, 6, 743–775.

RITTER, F. E. AND LARKIN, J. H. 1994. Using process models to summarize sequences of human actions. *Hum. Comput. Interact. 9*, 3, 345–383.

RITTER, F. E. AND MAJOR, N. P. 1995. Useful mechanisms for developing simulations for cognitive models. *AI Simul. Behav. Q. 91* (Spring), 7–18.

RITTER, F. E., JONES, R. M., AND BAXTER, G. D. 1998. Reusable models and graphics interfaces: Realising the potential of a unified theory of cognition. In *Mind Modeling—A*

Cognitive Science Approach to Reasoning, Learning and Discovery, U. Schmid, J. Krems, and F. Wysotzki, Eds. Pabst Scientific Publishing, Lengerich, Germany, 83–109.

RITTER, F. E., NERB, J., AND KINDSMÜLLER, M. 1994. Steps towards a series of models for a developmental task. In *Proceedings of the EuroSoar 8 Workshop*, K. Hurts and K. van Putten, Eds. University of Leiden, Leiden, 95–99.

SEARS, A. 1993. Layout appropriateness: A metric for evaluating user interface widget layouts. *IEEE Trans. Softw. Eng. 19*, 7 (July), 707–719.

SHERRILL-LUBINSKI CORPORATION. 1994. SL-GMS Technical Overview. Sherrill-Lubinski Corporation, Corte Madera, CA.

SZEKELY, P., LUO, P., AND NECHES, R. 1993. Beyond interface builders: Model-based interface tools. In *Proceedings of the ACM Conference on Human Factors in Computing* (INTERCHI '93, Amsterdam, The Netherlands, Apr. 24–29), S. Ashlund, A. Henderson, E. Hollnagel, K. Mullet, and T. White, Eds. ACM Press, New York, NY, 383–390.

TAATGEN, N. A. 1998. Explicit learning in ACT-R. In *Mind Modeling—A Cognitive Science Approach to Reasoning, Learning and Discovery*, U. Schmid, J. Krems, and F. Wysotzki, Eds. Pabst Scientific Publishing, Lengerich, Germany, 233–252.

VERA, A. H., LEWIS, R. L., AND LERCH, F. J. 1993. Situated decision-making and recognition-based learning: Applying symbolic theories to interactive tasks. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates Inc., Hillsdale, NJ, 84–95.

WOOD, D. AND MIDDLETON, D. 1975. A study of assisted problem solving. *Br. J. Psychol. 66*, 2, 181–191.

ZETTLEMOYER, L. S. AND ST. AMANT, R. 1999. A visual medium for programmatic control of interacive applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '99, Pittsburgh, PA, May), ACM Press, New York, NY, 199–206.

Received: June 1999; revised: November 1999; accepted: December 1999