In J. Carroll (Ed.), 2001. Human-computer interaction in the new millennium. 125-147. Reading, MA: Addison-Wesley.

User interface evaluation: How cognitive models can help

FRANK E. RITTER*,
Penn State University

GORDON D. BAXTER
University of York

GARY JONES
University of Derby

and

RICHARD M. YOUNG
University of Hertfordshire

* Contact author, School of Information Sciences and Technology, Penn State, 512 Rider Building, 120 S. Burrowes St., University Park, PA 16801-3857 (814) 865-4453 (office), 865-5604 (fax)
ritter@ist.psu.edu

Abstract

Cognitive models are computer programs that simulate human performance. 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. 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 several case studies showing how models can interact using mechanisms 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 UIMSs can and should be extended to create CMIMSs, and models can and should use CMIMSs to look at larger and more complex tasks. CMIMSs will help to exploit the synergy between the disciplines of cognitive modeling and HCI by supporting cognitive models as users.

Support has been provided by DERA and by the UK ESRC Centre for Research in Development, Instruction and Training. The views expressed in this paper are those of the authors and should not be attributed to the UK Ministry of Defence. In addition to the authors and those cited, 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 thank Jans Aasman, Erik Altmann, Paul Bennett, Michael Byrne, Wayne Gray, Steve Sawyer, and several anonymous reviewers for comments.

1. THE SYNERGY BETWEEN COGNITIVE MODELING AND HCI

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.

We describe here an approach for allowing cognitive models[1] more direct access to the interfaces users see. This is done by adapting ideas taken from user interface management systems (UIMSs) and extending them 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 (e.g., Olsen, Foley, Hudson, Miller, & Meyers, 1993). There are advantages of this approach for HCI and for modeling.

1.1. The Advantages for HCI

Human-computer interaction (HCI) has used cognitive models successfully in three main ways (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 GOMS family of techniques (Card, Moran, & Newell, 1983; John & Kieras, 1996) particularly have been successfully deployed. These models can help create and choose better designs, sometimes saving millions of dollars (e.g., Gray, John, & Atwood, 1993). The next step is to provide 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, Corbett, Koedinger, & Pelletier, 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 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 & Mavor, 1998), for example, to simulate fighter aircraft crews in a simulated war scenario (Jones, Laird, Nielsen, Coulter, Kenny, & Koss, 1999). In the future they will also lead to models that can test interfaces by behaving like a user. Using models as users has been envisioned before (e.g., Byrne, Wood, Sukaviriya, Foley, & Kieras, 1994; Lohse, 1997), but

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

has not yet been widely applied. The next steps are to provide models with more realistic inputs and outputs mirroring human performance and to apply them more widely.

A real impediment to each of these uses has been the difficulty of connecting the cognitive models to their task environment (Ritter & Major, 1995). Either the connection or the interface has to be built -- sometimes both. Further, this 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 to interaction with an external world (for a review of early models, see Ritter & Larkin, 1994). If previous models (e.g., Bauer & John, 1995; Howes & Young, 1996; John, Vera, & Newell, 1994) had been able to use the same interface as the corresponding subjects used, they would have been applied and tested on more tasks and would have been able to cover a wider range of behavior, skills, and knowledge.

Creating an model that is embodied (i.e., with perception and motor actions) further constrains a model by restricting the model to interact only through its hand and eye. 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 later incorporate knowledge related to interaction such as where to look on the interface to find information. The models require a depth and range of knowledge about interaction, predicting that users do too.

Keeping the task simulation distinct from the cognitive model has three advantages. First, it makes development easier because the model and the simulation can be tested and debugged independently. Second, it makes it less likely that the modeller 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.

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, which is a

difficult activity. There is less development required because only one interface has to be created and updated if the model (or others) suggest changes. There is also less question about whether the model and the subject had access to the same material. Several of the case studies use simulations that already existed, thus relieving the modeler from developing the task simulation.

Finally, this approach leads 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 & 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 & Lebiere, 1998). Cognitive architectures offer a platform for developing cognitive models rapidly whilst still maintaining theoretical coherence between the models.

Although there are tools to develop user interfaces, and there are tools that help develop cognitive models, there are none that support connecting cognitive models to a wide range of interfaces. In the rest of this paper 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 examples where models perform interactive tasks using the same type of simulated eye and hand implemented in different interface tools and modeling languages. These examples, when considered together with a review of related systems, suggest possible applications and indicate where further work is necessary. Section 4 assesses the implications of these projects and identifies ways in which integrated models could be exploited in the development of user interfaces.

2. A ROUTE TO SUPPORTING MODELS AS USERS

The cognitive modeling process is unique in many respects, although the artifacts created by it are similar 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 examined 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 & Larkin, 1994).

The cognitive modeling process can be viewed as producing three artifacts, each of which fulfills a particular purpose. The first artifact is the cognitive model itself, which simulates the cognitive performance and behavior of a human performing the task. As theory, it has primacy.

The second artifact is a task application or its simulation. 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, can often be implemented using the cognitive modeling language. Dynamic tasks, however, where the state of the environment can evolve without outside intervention, are 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 model and the task simulation. It simulates human perception and action and provides a way for the model and simulation to communicate. 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.

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 good candidates to build upon.[2]

2.2 The Role of User Interface Management Systems

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 can be modified to approximate human limitations and capabilities. Any initial considerations for a toolkit to support the cognitive modeling process will therefore need to incorporate (a) a tool to create interfaces, (b) a run-time mechanism that lets the cognitive model interact with the task simulation (i.e., a model eye and hand), and (c) a communication mechanism that passes information between the cognitive model and the task simulation.

User Interface Management Systems (UIMSs) provide a similar set of features for the development of interactive applications, supporting interface creation and helping to manage the

[2]An alternative linkage mechanism is to recognize objects directly from the screen (Zettlemoyer & St. Amant, 1999).

interaction when the interface is used (e.g., Myers, 1995). UIMSs can be used to create interfaces and applications in their implementation language, or can create interfaces that are tied to external applications. By definition UIMSs provide a set of features that very closely match our requirements.

UIMSs 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 UIMSs, particularly the parallel needs between (human) users and cognitive models.

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. UIMSs provide facilities that support these functional capabilities. In particular, in UIMSs there are tools to find which objects occlude other objects (such as the simulated eye being over a textual label on the display), 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, also 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 UIMSs. The dashed extension to the UIMS arrow denotes that the facilities needed to implement the eye and hand can usually be based on existing facilities in the UIMS. However, the simulated eye and hand place requirements that not all UIMSs or related systems currently satisfy.

Next, the CMIMS arrow reflects our suggestion for the development of Cognitive Modeling Interface Management Systems. A CMIMS would need to include the simulated eye and hand, as well as the communication mechanism between them and the cognitive model.

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, however, visual search is required to find information to examine, the amount of information available at any point is limited, and performing visual search takes time. Similar restrictions apply to the motor output.

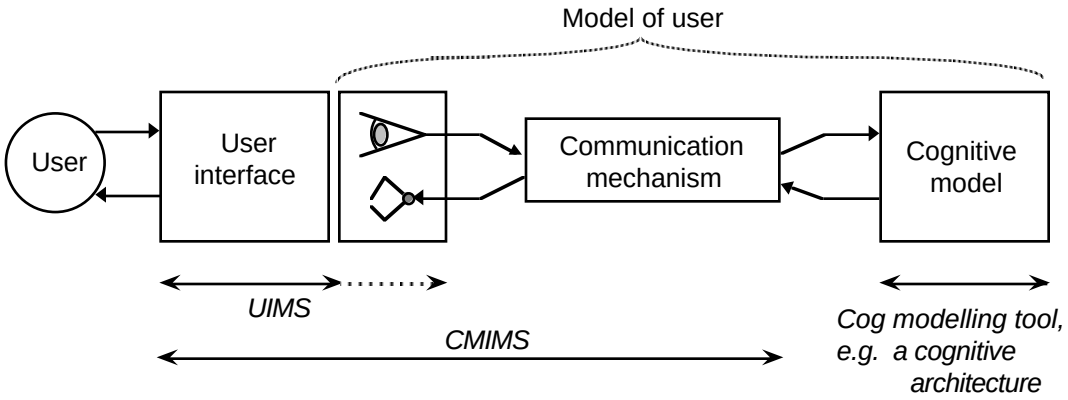

Fig. 1. A cognitive model tied to a user interface of a task simulation, where the model and the simulation may be running in different environments (programming languages, processes, and/or computers). The hand and eye are implemented in the same UIMS as the task simulation.

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.

2.4 A Functional Model Eye and Hand

The Sim-eye and Sim-hand[3] are the most important parts of the CMIMS. They bring the model into contact with the interface. We have created an initial set of functional capabilities and empirical regularities on which to base models of interaction (Ritter, Baxter, Jones, & Young, in press). With each additional refinement to the implementation, the models had more capabilities and have exhibited more of the regularities.

These capabilities and regularities, such as the size of visual acuity and the speed of motor movements were chosen based on a literature review (Baxter & Ritter, 1996) as important for an initial model of interaction. These capabilities are fundamental necessities to support interaction, and the regularities are the most important constraints on performance.

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. The most important point is first to support the process of providing cognitive models access to interfaces in UIMSs by providing these functional capabilities. 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 uses these observations to refine the implementations and use of the Sim-eye and Sim-hand.

The Sim-eye and Sim-hand are controlled by the model through a simple command language. 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 and the model see the same display (to the limit of the theory of vision implemented in the Sim-eye).

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

[3] In order to emphasise the distinction between the model's capabilities and human capabilities, we 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.

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 case studies show that it is possible to create CMIMSs, and the advantages in so doing.

3. EXAMPLE COGNITIVE MODELS THAT INTERACT

We have created a series of cognitive models that interact with task simulations. We present here examples developed using tools that can be described as UIMSs. Two further examples are available that use a Tcl/Tk based CMIMS to model dialing a variety of telephones (Harris, 1999; Lonsdale & Ritter, 2000) and exploratory search in interfaces (Ritter et al., in press). 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 A Simplified Air Traffic Control Model

The first task simulation is a simplified air traffic control (ATC) task (Bass, Baxter, & Ritter, 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. Such a model could also help support the user by predicting what they would do, and then to assist them or do it for them.

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 is 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 aircraft model. 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 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 finding and 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. A crucial element of the task is that 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, Giuse, Dannenberg, Vander Zanden, Kosbie, Pervin, et al., 1990), chosen because it was familiar and provides fairly general support for creating interfaces. The model was implemented in Soar.

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. 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 the location of objects that appear outside the fovea. The Sim-eye is moved around the ATC display window (center left in Figure 2) by the model placing saccade commands to be processed into the Soar-IO facility. When the Sim-eye saccades, previous visual elements are removed from cognition, a plausible theory of visual input (Horowitz & Wolfe, 1998).

The Sim-hand was initially implemented as simple function calls to the task simulation via the Soar-IO facility. Later, a Sim-hand (Rassouli, 1995) was included. The revised version of the ATC cognitive model helped to illustrate two of the difficulties involved in modeling task interaction. (a) 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 (b) 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.

3.1.2 The communication mechanism. The communication mechanism is implemented by a system called MONGSU (Ong, 1994), based on UNIX sockets. It allows any pair of Lisp or C-based processes to communicate using list structures and attribute-value pairs.

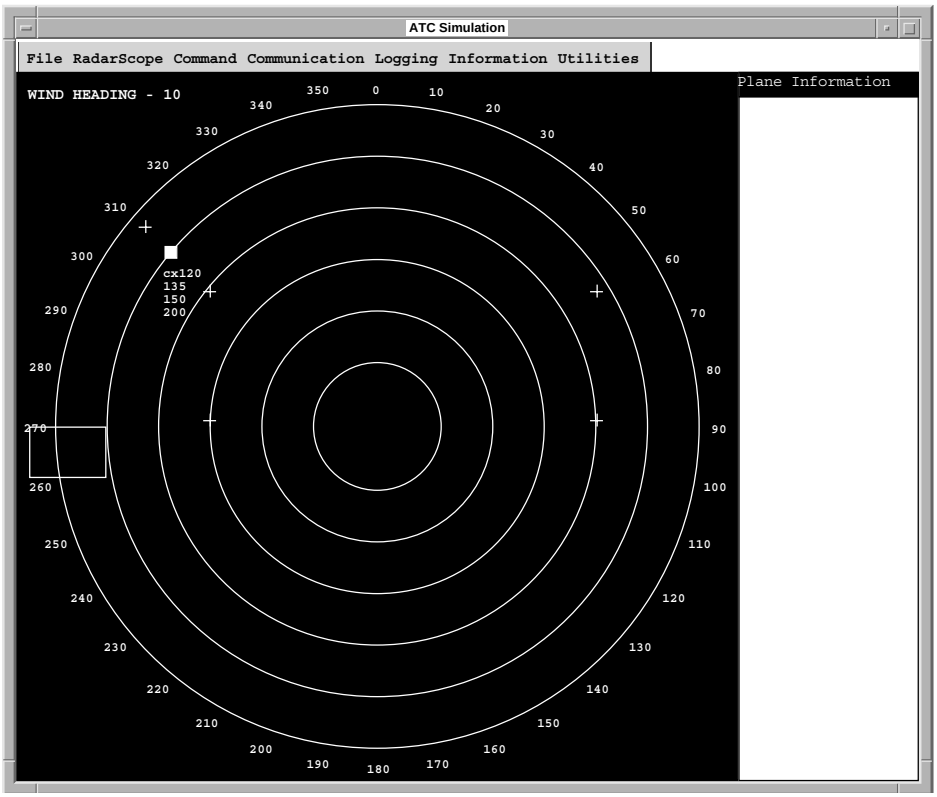

Fig. 2. The ATC simulation display showing the model's fovea (the white rectangle on the left hand side of the figure), before it moves to the plane labeled "cx120".

3.1.3 Summary. The ATC model demonstrated that a simple but functional Sim-eye could be created using an existing UIMS. This model used knowledge that has not often been seen in a cognitive model: where to look and how to visually monitor. Knowledge about the ATC display had to be included in the cognitive model because when and where to look at the screen is domain dependent.

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. 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, otherwise the model has tunnel vision and must carefully scan the entire screen with its fovea. So, an apparently trivial task became an intricate process, involving its own level of problem solving and search based on the external interface. 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.

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.

The use of sockets as a communication mechanism added an extra layer of complexity because it required the modeller 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.

3.2 Tower of Nottingham Model

The Tower of Nottingham is a puzzle where wooden blocks are assembled to form a pyramid of 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.

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 model's and subjects' visual search and object manipulations in service of problem solving are analogous to manipulation in graphical user interfaces and to some aspects of graphic design in drawing and CAD/CAM packages.

The blocks simulation is written in Garnet. The complexity of the task precludes simulating the task with a cognitive modeling language. The model is written using the ACT-R (v. 3.0) cognitive architecture (Anderson & Lebiere, 1998), although an earlier model was in Lisp.

3.2.1 Visual Perception and Motor Action. The task simulation, shown in Figure 3, includes a Sim-eye and a pair of Sim-hands represented graphically, which makes its behavior readily visible. The Sim-eye and Sim-hand are controlled using commands similar to the ATC system's commands, 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, release, and 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.

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. 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 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.

Once an action is completed the cognitive model is informed. The model can then manipulate the Sim-eye to verify the results. 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.

The model thus 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 abandoned. This behavior of moving hands to blocks but not picking them up seems to occur in adults, and suggests there are types of mistakes that are not fully overt. This type of mistake is likely to occur for adults using interfaces as well.

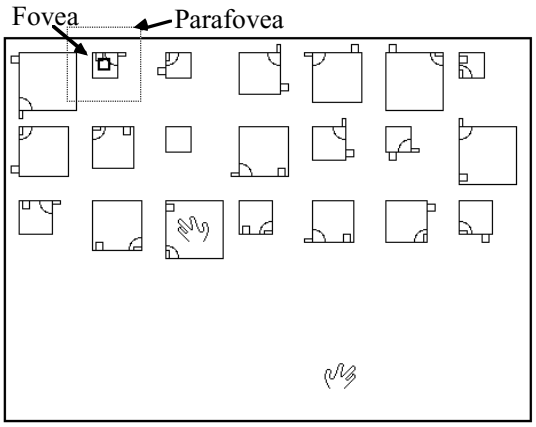

Fig. 3. The Tower of Nottingham. The fovea is the small black outlined square in the center of the block in the top left corner. The parafovea, shown as a dashed line, extends approximately two fovea widths in each direction beyond the fovea. The left Sim-hand has picked up one of the largest blocks.

3.2.2 The Communication Mechanism. Interaction between the cognitive model and the task simulation uses Lisp function calls because both are in 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 noted as achieved. The model can then terminate the goal and continue its behavior.

3.2.3 Summary. This case study further indicates that 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 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 performance of the model matches the performance of adult subjects on the task reasonably well because the whole task was modeled and the necessary learning could occur in visual search, in cognition, and in output. 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.

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 changes in strategy choice is the most likely candidate for what develops in children (Jones, Ritter, & Wood, 2000).

3.3 Electronic Warfare Task Model

The final example is a simulated eye and hand in SLGMS, an object oriented, real-time dynamic UIMS. Tenet Systems, the UK sales agency for SLGMS, developed the Sim-eye and Sim-hand under our direction. They had access to the SLGMS system's source code, which is necessary for developing a CMIMS. The design, and to the best of our knowledge the implementation, allows the model to interact with any SLGMS interface.

In the upper window of Figure 4 there is a simplified interface that was used to develop and test the Sim-eye and Sim-hand. In the lower window is a control panel based on what we learned from the Tcl/Tk models (Ritter et al., in press). The control panel displays the current state of the Sim-eye and Sim-hand, making debugging easier because the state it visible and the Sim-eye and Sim-hand can be manipulated prior to programming the model.

3.3.1 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, 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 commands to manipulate the Sim-eye and Sim-hand are the same as in previous examples.

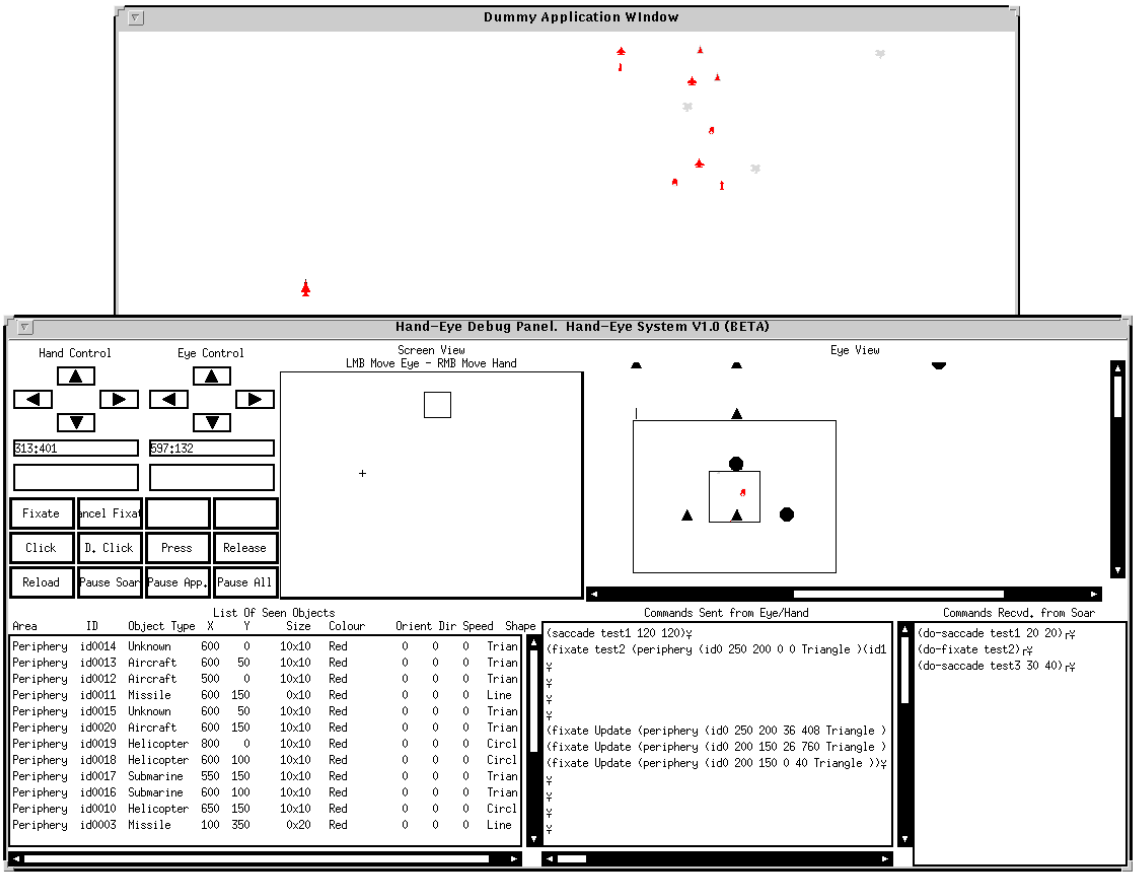

Fig. 4. The Dummy Application Window is a simple test application. The control panel includes controls (upper left of this window) for the Sim-eye and Sim-hand, and continuing along its top, two displays showing the current position (screen view) and detailed contents of the fovea (eye view). Along the bottom are listings of objects sent from the eye to the model, and the commands in each direction sent through the linkage mechanism. Normally these two windows are on different monitors.

3.3.2 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 example. 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.

3.3.3 Summary. We have 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, demonstrating that CMIMSs can be quickly understood and implemented by others.

Using two separate display screens, with the simulation (driven by the cognitive model) on one screen and the control panel (driven by the modeller) on the other, solves several problems. During debugging there is a need for the modeller 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 additional debugging facilities. Using separate screens allows the cognitive model to control the Sim-hand's mouse pointer on one screen, whilst the modeller has control of the mouse pointer for the control panel on the other screen. Without this capability the modeller cannot query the model while it runs and the model and modeller come into conflict trying to use the same mouse.

An initial model of a corresponding radar task with this display 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 interface in Figure 4 will profoundly change the model's task from a geometry task of computing intercept angles to a scanning and monitoring task with geometry as only a subcomponent.

3.4. Related Systems

Other examples where models interact with simulations provide further lessons. The related systems presented here have used three UIMSs, and typically can present displays on more than one type of machine. The simulation languages are usually closely tied to the UIMS.

The Soar agent simulations (Jones et al., 1999), which support large scale military training exercises, and MIDAS (Corker & Smith, 1993), which is a design tool for aviation systems that includes a model user, 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. 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.

The models and simulations in APEX and Driver-Soar illustrate some of the possible applications and results that are available from employing models as surrogate users. APEX (Freed & Remington, 1998) is a design tool to predict errors in complex domains. It has been applied to air traffic control systems and cockpits. It provides cognitive models in its own architecture, simulated eyes and hands, and a communication mechanism between them. It is not based on a UIMS but modelers 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. It is limited in the types of tasks that can be examined, however.

Driver-Soar (Aasman & Michon, 1992) is a detailed model of driving that interacts with a 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.

EPIC (Kieras & Meyer, 1997) is a system addressing the intricate details of perception and action. 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. 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. Later versions access a visual display shared with users. EPIC can be used to make accurate predictions of interaction behavior such as menu use. Some of EPIC's capabilities and the regularities they are support have been used by Soar models (Chong & Laird, 1997) and by ACT-R/PM (Byrne & Anderson, 1998).

The only other system that could properly be described as a CMIMS is ACT-R/PM (Byrne, 1999). 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 has been demonstrated through multiple use by models (Byrne, 1998; multiple examples in Taatgen & Aasman, 2000).

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 modeller 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 & Anderson, 1998). However, it can already recognize most objects and 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 to a good UIMS but not a common or widely portable UIMS.

Some of these systems are more accurate and allow cognition and interaction to occur in parallel. Often, they have not put development effort into their own usability and have not used more general UIMSs (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 accuracy or the model's own usability are unimportant.

3.5. Limitations of This Approach

There are several limitations to the current generation of systems that could be classed as CMIMSs. 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.

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, Luo, & Neches, 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, Schoelles, & Fu, 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 AS USERS IN THE NEW MILLENNIUM

Supporting cognitive models as surrogate users is possible. The case studies show that 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 routinely apply theoretically grounded cognitive models to real world HCI tasks. In this section we review the approach, noting how it can contribute to the development of cognitive models and what it means for the future of interfaces.

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 UIMSs from the same design. This reimplementation of the same approach provides a form of constraint because it the design is reused and the capabilities and regularities are noted explicitly. A more important aspect is that the Sim-eye and Sim-hand are now available in several widely used software tools, so reuse should be a possibility in the future.

Modelers should use these Sim-eyes and Sim-hands to provide their models with an interactive capability. ACT-R/PM's hand is available at www.ruf.rice.edu/~byrne/RPM/; the Tcl/Tk eye/hand will be available at ritter.ist.psu.edu. Similarly, newer cognitive architectures should provide at least one CMIMS for their models.

Authors of interface design tools and of UIMSs should include support for cognitive models as users. The functional capabilities and experimental requirements noted here and in related documents (Baxter & Ritter, 1996; Ritter et al., in press) 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 UIMSs.

4.1. Implications for Models

The models presented here would not have been possible to develop without access to external simulations. The models are far more complex because they interact with tasks, tasks too complex to simulate in a cognitive modeling language.

Including a theory of interaction has both provided models with more capabilities and also constrained the speed and abilities of the models, in a way approximating human behavior. CMIMSs provide a way to encapsulate these constraints. Interacting has required adding new knowledge and new types of knowledge to the models, including where to look and what to do with what it sees. This result suggests that there are types of knowledge (i.e., visual 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 using 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. This approach thus helps make more complete cognitive architectures.

The case studies 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, Lonsdale & Ritter, 2000). Models as well as users can work with the same interface (e.g., the ATC task). This

approach will contribute to the reuse of models envisioned for cognitive architectures (Newell, 1990) and provide a further constraint on the architecture.

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 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 people and Soar could use, and one that has numerous existing interfaces and tasks.

4.2. Implications for Interfaces

There are at least two significant ways in which CMIMSs can be used 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. Models of phone interaction are starting to be able to do this (Lonsdale & Ritter, 2000). It becomes possible to dynamically evaluate *how* an interface is used, and where important events like errors may occur (Freed & Remington, 1998). Cognitive models of interface use, such as IDXL (Rieman, Young, & Howes, 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 in hand, it will allow more time to be spent developing the models. Models as 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 UIMSs 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 come through supporting cognitive models as users, but it will provide a new way to test interfaces.

REFERENCES

Aasman, J., & Michon, J. A. (1992). Multitasking in driving. In J. A. Michon & A. Akyürek (Eds.), *Soar: A cognitive architecture in perspective.* Dordrecht, The Netherlands: Kluwer.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167-207.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Lawrence Erlbaum.

Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Creating cognitive models to control simulations of complex systems. *AISB Quarterly, 93*, 18-25.

Bauer, M. I., & John, B. E. (1995). Modeling time-constrained learning in a highly interactive task. In I. R. Katz, R. Mack, & L. Marks (Eds.), *Proceedings of the CHI '95 Conference on Human Factors in Computer Systems*. 19-26. New York, NY: ACM SIGCHI.

Baxter, G. D., & Ritter, F. E. (1996). Designing abstract visual perceptual and motor action capabilities for use by cognitive models (Tech. Report No. 36). ESRC CREDIT, Psychology, U. of Nottingham.

Byrne, M. D. (1999). ACT-R Perceptual-Motor (ACT-R/PM) version 1.0b5: A users manual. Houston, TX: Psychology Department, Rice University. Available at <http://www.ruf.rice.edu/~byrne/RPM/project.html>.

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

Byrne, M. D., Wood, S. D., Sukaviriya, P., Foley, J. D., & Kieras, D. E. (1994). Automating interface evaluation. In *Proceedings of the CHI '94 Conference on Human Factors in Computer Systems*. 232-237. New York, NY: ACM.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chong, R. S., & 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*. 107-112. Mahwah, NJ: Lawrence Erlbaum.

Corker, K. M., & Smith, B. R. (1993). An architecture and model for cognitive engineering simulation analysis: Application to advanced aviation automation. In *Proceedings of the AAIA Computing in Aerospace 9 Conference*. . San Diego, CA: AIAA.

Freed, M., & Remington, R. (1998). A conceptual framework for predicting error in complex human-machine environments. In M. A. Gernsbacker & S. J. Derry (Eds.), *Proceedings of the 20th Annual Conference of the Cognitive Science Society*. 356-361. Mahwah, NJ: LEA.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8*(3), 237-309.

Gray, W. D., Schoelles, M., & Fu, W.-T. (1998). When milliseconds matter: Implementing microstrategies in ACT-R/PM. In *Proceedings of the 5th ACT-R Workshop*. 130-135. Psychology Department, Carnegie-Mellon University.

Harris, B. (1999). *PracTCL: An application for routinely tying cognitive models to interfaces to create interactive cognitive user models.* BSc thesis, Psychology, U. of Nottingham.

Horowitz, T. S., & Wolfe, J. M. (1998). Visual search has no memory. *Nature, 357*, 575-577.

Howes, A., & Young, R. M. (1996). Learning consistent, interactive, and meaningful task-action mappings: A computational model. *Cognitive Science, 20*(3), 301-356.

John, B. E. (1998). Cognitive modeling for human-computer interaction. In *Proceedings of Graphics Interface '98*. 161-167.

John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction, 3*(4), 287-319.

John, B. E., Vera, A. H., & Newell, A. (1994). Towards real-time GOMS: A model of expert behavior in a highly interactive task. *Behavior and Information Technology, 13*, 255-267.

Jones, G., Ritter, F. E., & 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., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

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.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*(1), 1-64.

Lohse, G. L. (1997). Models of graphical perception. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), *Handbook of Human-Computer Interaction.* 107-135. Amsterdam: Elsevier Science B. V.

Lonsdale, P. R., & Ritter, F. E. (2000). Soar/Tcl-PM: Extending the Soar architecture to include a widely applicable virtual eye and hand. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the 3rd International Conference on Cognitive Modelling*. 202-209. Veenendaal (NL): Universal Press.

Myers, B. A. (1995). User interface software tools. *ACM Transactions on Computer-Human Interaction, 2*(1), 64-103.

Myers, B. A., Giuse, D. A., Dannenberg, R. B., Vander Zanden, V., Kosbie, D. S., Pervin, E., Mickish, A., & Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer, 23*(11), 71-85.

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

Olsen, D. R., Foley, J. D., Hudson, S. E., Miller, J., & Meyers, B. (1993). Research directions for user interface software tools. *Behaviour & Information Technology, 12*(2), 80-97.

Ong, R. (1994). *Mechanisms for routinely tying cognitive models to interactive simulations*. MSc thesis, U. of Nottingham. Available as ESRC Centre for Research in Development, Instruction and Training Technical report #21 and as ftp://ftp.nottingham.ac.uk/pub/lpzfr/mongsu-2.1.tar.Z.

Pew, R. W., & Mavor, A. S. (Eds.). (1998). *Modeling human and organizational behavior: Application to military simulations*. Washington, DC: National Academy Press. http://books.nap.edu/catalog/6173.html.

Rassouli, J. (1995). *Steps towards a process model of mouse-based interaction*. MSc thesis, Psychology, U. of Nottingham.

Rieman, J., Young, R. M., & Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. *International Journal of Human-Computer Studies*, 743-775.

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

Ritter, F. E., & Larkin, J. H. (1994). Using process models to summarize sequences of human actions. *Human-Computer Interaction, 9*(3), 345-383.

Ritter, F. E., & Major, N. P. (1995). Useful mechanisms for developing simulations for cognitive models. *AISB Quarterly, 91*(Spring), 7-18.

Sears, A. (1993). Layout appropriateness: A metric for evaluating user interface widget layouts. *IEEE Transactions on Software Engineering, 19*(7), 707-719.

Szekely, P., Luo, P., & Neches, R. (1993). Beyond interface builders: Model-based interface tools. In *Proceedings of InterCHI '93*. 383-390. New York, NY: ACM.

Taatgen, N., & Aasman, J. (Eds.). (2000). *Proceedings of the 3rd International Conference on Cognitive Modelling*. Veenendaal (NL): Universal Press.

Zettlemoyer, L. S., & St. Amant, R. (1999). A visual medium for programmatic control of interactive applications. In *CHI '99, Human Factors in Computer Systems*. 199-206. New York, NY: ACM.