Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Creating Models to Control Simulations: A Generic Approach . *AI and Simulation of Behaviour Quarterly, 93,* 18-25.

# Creating Models to Control Simulations: A Generic Approach

*Ellen J. Bass*
*Center for Human-Machine Systems Research*
*Georgia Institute of Technology*
*Atlanta, GA 30332-0205, USA*
*ellen@chmsr.gatech.edu*

*Gordon D. Baxter, Frank E. Ritter*
*Departments of Psychology and Computer Science*
*University of Nottingham*
*Nottingham NG7 2RD, United Kingdom*
*gdb, ritter @psyc.nott.ac.uk*

## ABSTRACT

When things go wrong with complex systems, the consequences can be disastrous. Computer simulations are often used for training operators to control these systems and as a way of safely investigating system behaviour. If models of human operators can be used to drive these simulations, this should help to highlight possible system design problems in a more cost-effective manner than having to use real operators. We are developing a generic approach for creating cognitive models that can learn to control any simulation that has its user interface written using a particular toolkit. This approach is based on using a robust cognitive architecture and a robust interface development tool. We identify a set of general requirements that the models should meet. An initial model, which satisfies a number of these, controls a simulation of a simple air traffic control task. The model learns while performing the task interactively and by reflecting on task performance afterwards.

## KEYWORDS

complex systems, human-computer interaction, simulation, Soar, user modelling

## INTRODUCTION

Interactive computer simulation provides a means of training operators and for investigating behaviour of complex systems without the drawback of potentially disastrous consequences arising when things go wrong. In situations where human operators must be simulated, cognitive models (i.e., information processing algorithms that perform the task with behaviour similar in sequence and form to humans performing the same task) can take their place (e.g., Tambe, Johnson, Jones, Koss, Laird, et al., 1995). If these models[1] are sufficiently accurate, they can not only control the simulation, but also be used to make predictions about human operator performance (Card, Moran, & Newell, 1983), about potential operator errors (e.g., Sasou, Takano, Yoshimuraa, Haraokab, & Kitamurac, 1995; Young & Whittington, 1990), and about required operator support and training (e.g., Knaeuper and Morris, 1984).

Developing models of human operators of complex systems is difficult. The theory needed to create such models is incomplete because research in artificial intelligence and cognition has tended to focus on laboratory tasks using situations not generally comparable to the real world. In cases where the work has addressed real world settings, problems of generalisation and of transfer of the results to other domains arise. Integrating the relevant models would require significant effort, and the resulting models would still be incomplete.

Once the model has been built, getting it to interact with the simulation introduces another level of complexity. In some cases where models have been built to control a simulation, their communication has not been straightforward. One typical problem is that simulations and models are often developed on different types of machines using different software. This means that the communication links between the model and the simulation must be hand-crafted, offering little transfer between efforts (e.g., Pearson, Huffman, Willis, Laird, & Jones, 1993).

Even when the model can communicate with the simulation, the problem of ensuring the psychological plausibility of the interaction between them remains. Since most simulations are controlled via a user interface, and decision making can be affected by perception and motor action (Suchman, 1987; Hammond, Stewart, Brehmer, & Steinman, 1975), it follows that models controlling simulations should incorporate these influences. For example, in order to perceive all of the data on the display, the model must take into account the physical limits of vision, and hence only clearly perceive a small amount of the information on the screen at any one time. The EPIC architecture (Kieras, Wood, & Meyer, 1995) encapsulates these ideas. Freed and Johnston (1995) are also developing such a model that is being used to help predict the impact of changing technology on performance and error rates of air traffic controllers.

[1]Unless specified, “model” refers to “cognitive model” and "simulation" to "simulation of a complex system".

Our approach fits into the field of what has been called modelling of macro-cognition (Cacciabue & Hollnagel, 1993). This puts it in the same category as work on models such as COSIMO (Cacciabue, Decortis, Drozdowicz, Masson & Nordvik, 1992), CES (Woods, Roth & Pople, 1987, cited in Cacciabue & Hollnagel, 1993), AIDE (Amalberti & Deblon, 1992), and the work of Freed and Johnston (1995). Our approach differs from these in that it is based on an existing robust theory of cognition that can learn: Soar (Laird, Newell, & Rosenbloom, 1987). Also the implementation of our approach is designed to apply to all applications using the same user interface toolkit. When this toolkit is used, the resulting user interface can be used by operators and the model.

## REQUIREMENTS FOR COGNITIVE MODELS THAT CONTROL SIMULATIONS

Taking into account the goal of using models to control simulations for training and investigating system behaviour, we can list several requirements that these models should meet. This is not a complete list, but is, we believe, a substantial one. All of these requirements will not be met by our initial model, but they indicate where to focus our efforts.

*Models must run in real-time.* Timing is critical in dynamic domains. Realistic models should perform at the same rate as a human operator with a comparable level of expertise. It is the pace of the simulation (i.e., the rate at which the simulated world changes) and the task to be accomplished that influence the rate at which the model should respond.

*Models must have general problem solving skills.* General reasoning and problem solving become important when controlling real systems because such systems involve too much uncertainty for all of the possible decision-making situations to be enumerated a priori. Just as human operators can bring general problem solving methods to bear when trying to accomplish the task at hand, models must also have these general skills. This makes the model flexible and able to adapt to unanticipated events.

*Models must be able to acquire new cognitive skills and new knowledge.* Learning is an important feature of any model that is intended to mimic behaviour of real operators. Operators apply and enhance domain and interface knowledge at various levels of abstraction while accomplishing their tasks (Rasmussen, 1985). Models should therefore be able to mimic how operators acquire various levels of knowledge and skill. The analyst can then investigate tailoring the simulation's interface, training, and on-line help to assist this learning process.

Models should also be able to request and to take instruction if this functionality is required by the interface (i.e., if the simulation includes an embedded assistant) or the model is collaborating with other operators. This capability is helpful in allowing models to fill in incomplete or incorrect knowledge and to note where such knowledge deficiencies occur.

The time taken to perform a task decreases with practice (Newell & Rosenbloom, 1981). This speedup can also be influenced by strategy changes (Reder & Ritter, 1992). If models are to learn and to exhibit these effects, they must incorporate multiple performance strategies. They also need to be able to learn new strategies and have the ability to switch between them.

*Models must be able to make errors.* It is generally difficult to define precisely what constitutes an operator error. This is because any errors that arise in a human-machine system can often only be categorised as human error after they have occurred and their cause has been analysed. Operator errors therefore play an important part in the design of any man-machine system. This means that in order to test the validity of models of human performance, error behaviour must be analysed, since, as Rasmussen (1986) states:

> Success in prediction and simulation of well-adapted human interaction is no proof of the validity of a model; only analysis of situations when adaptation breaks down will be informative, inasmuch as the aspects of human resources and their limitations must be reflected by a model. (p.149)

Although many models have been developed for error-free behaviour, modelling typical patterns of error behaviour could help to identify appropriate operator support during real-time operations and requirements for operator training.

*Models must include domain-specific cognitive aspects in order to mimic expert performance.* Models must be able to utilise domain-specific knowledge to control the simulation. This capability, included in all knowledge-based expert systems, is a necessary, but not a sufficient, requirement. With domain-specific knowledge, the model of expert behaviour is a normative one that can be used to not only control the simulation but also to indicate which data should be displayed in order to perform the task (Mitchell & Saisi, 1987), sub-task frequencies, and typical and necessary sub-task sequences (Card, et al. 1983).

*Models must have perceptual ability.* Models must be able to perceive the information available from the simulation via the user interface. The level of perceptual ability should be sufficient to allow the model to exhibit the regularities documented in the literature (e.g., Wade & Swanston, 1991). For example, a simulated eye must include a fovea[2] and be able to saccade[3] in psychologically plausible ways to obtain information from the display. For objects outside of the foveal region, the model should only have access to limited information, such as object location. Typically, models perform the task more quickly than people and the lack of psychologically plausible perception is likely to contribute to this difference.

*Models must have motor skills.* Motor action must be treated in a similar way to perception: the model must explicitly interact with the simulation. The motor action should conform to the appropriate regularities (e.g., Fitts' Law (1954)). As an approximation, the timing limitations can be based on analyses for the domain of interest using the Keystroke Level Model (Card, Moran, & Newell, 1980).

*Models must have the knowledge to be able to use the interface.* A model should have some basic knowledge of human-computer interaction (HCI) (e.g., that a mouse is movable). It should be possible for the model to augment this knowledge by learning through its interaction with the simulation's interface. Earlier work has already investigated the development of models that learn while performing interactive tasks (e.g., Howes & Young, 1991; Rieman, Lewis, Young, & Polson, 1994). By extending these models, it should prove possible to make them applicable to more complex, knowledge intensive domains such as air traffic control (ATC).

Many systems now incorporate on-line help and manuals as part of the interface. Simulations will therefore also include these features so the controlling model must be able to process such information as part of the interaction process.

*Models should interact with the same interface as the human operator.* Figure 1 illustrates possible relationships between models, interactive task simulations, and human operators. In the first column, two separate interfaces and application simulations are created (one for the model and one for the human operator). This approach has been used by some (e.g., Peck & John, 1992) because it can be difficult, or impossible, for the model to communicate with the same application as the human operator.

The middle column of Figure 1 shows a better alternative where the model and the operator interact with the same world, using the same interface. This helps to ensure (but does not guarantee) that the model encounters the same features as the operator. It also allows the analyst to validate predictions from the model's performance against data from human subjects since the operator and the model are interchangeable. This approach can be used when the model runs in an environment compatible with the simulation's. The third column illustrates taking this approach one step further: the cognitive model becomes an embedded assistant which can aid the operator.

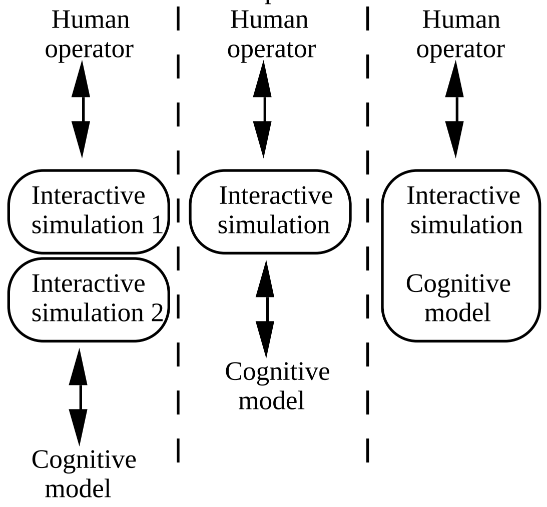

**Figure 1.** Possible approaches for a cognitive model and a human operator to control simulations.

## Research Approach

We are developing a generic approach which allows models to control simulations of complex systems based on the identified requirements. This approach begins to address the difficulties involved in modelling the knowledge and skills required to control such systems. We are initially focusing on cognitive decision-making and learning within the task domain, and on physical (i.e., eye and hand) interaction with the simulation. We are also developing a mechanism for learning through reflection.

We are setting up an environment to demonstrate the principle illustrated in the middle column of Figure 1. This is shown in more detail in Figure 2. The simulated operator on the left of the figure consists of a cognitive model and high level models of perception and motor action. This allows the simulated operator to control the simulation using the same user interface as the real operator shown on the right of the figure. The cognitive model not only reasons about the task, but also generates commands to control the simulated eye in order to receive perceptual input. It also sends commands to a simulated hand to perform motor actions to manipulate the interface.

## Current implementation

To instantiate the environment, we use a cognitive modelling architecture, a simulation language, and a mechanism for communication between the two. After surveying the options (Ritter & Major, 1995), we chose Soar (Laird, et al., 1987) with

[2]The fovea is the area on the retina of greatest visual acuity.
[3]A saccade is a rapid displacement of the eye's gaze to a new location, during which retinal stimulation is smeared.

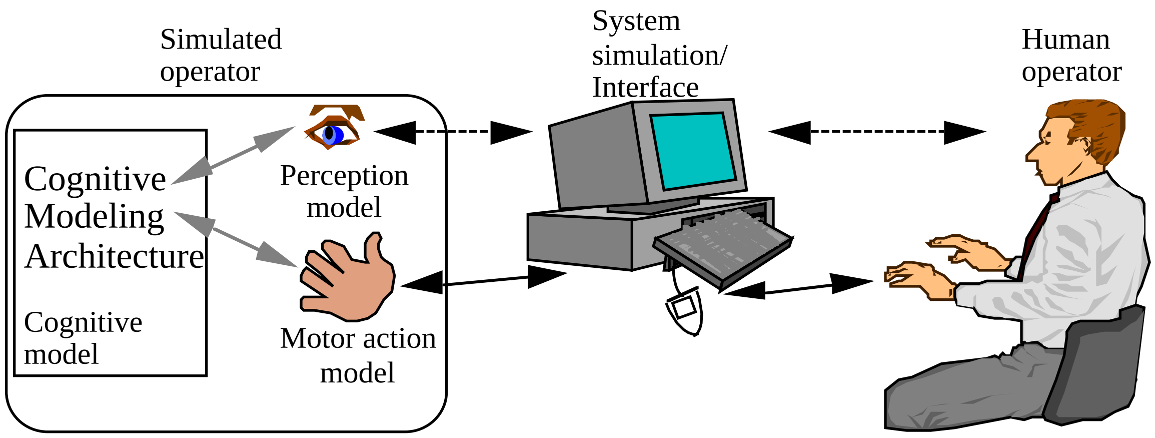

**Figure 2.** An operator model and a human operator interacting with a simulation via the interface.

its built in learning mechanism as the cognitive modelling architecture; Garnet (Myers, Guise, Dannenberg, Vander Zanden, Kosbie, et al., 1990) as the development language for the simulations of perception, motor action, the application and the user interface; and Unix-style sockets (e.g., Glass, 1993) for inter-process communication between the cognitive model and the simulations.

The combination of Soar and Garnet provide the ability to create broad agents with cognitive, perceptual, and motor capabilities. Soar permits rapid development of cognitive models that can learn. It allows for knowledge to be represented at varying levels of abstraction. Also existing Soar sub-models can be easily integrated. Garnet facilitates building user interfaces (and simulations to interact with them) because its library of user interface widgets[4] and functions to manipulate them are at a level of abstraction that is consistent with how interface designers and cognitive models think about them. Principled use of Garnet libraries and functions also provides a way to create reusable perceptual and motor capabilities.

We have selected a simplified ATC problem in order to illustrate our approach in a real domain. An initial version of the system allowed a simple model to communicate with the ATC simulation (Ong, 1994). This system continues to be refined and extended to incorporate high level models of perception and motor action that are controlled by the cognitive model.

## ATC simulation and its user interface

Our task involves selecting a runway and an arrival route for each landing aircraft and directing each aircraft along the chosen route to the runway. Aircraft are directed by issuing clearances to change speed, heading, and altitude.

Figure 3 depicts a portion of the airspace controlled in our simulation. There are two arrival routes and one runway. The runway is called Runway 9 when aircraft approach from the west and Runway 27 when they approach from the east. Landing aircraft that enter the airspace at entry fix E land by using Arrival Route 1 (where aircraft travel from fix E to D to A to B and land on Runway 27), depicted with dark arrows in Figure 3, or Arrival Route 2 (where aircraft travel from E to D to Q and land on Runway 9), depicted with dashed arrows. The model initially knows to land aircraft into the wind, which results in better landing performance. For example, if the wind is blowing from west to east, then aircraft entering at Fix E should fly along Arrival Route 1 and land heading to the west on Runway 27.

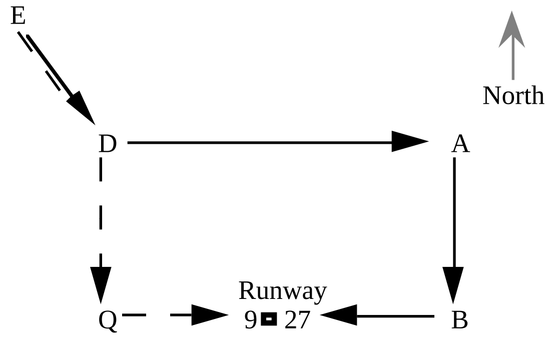

**Figure 3.** Arrival routes for aircraft entering at Fix E and landing on Runway 9/27.

Figure 4 shows a snapshot of the ATC simulation's display. In this instance, aircraft CX120 appears as a track symbol and a data

[4]A widget is a manipulable graphical object that is always mapped to the same specific function. Examples are radio buttons, scroll bars, and pull-down menus.

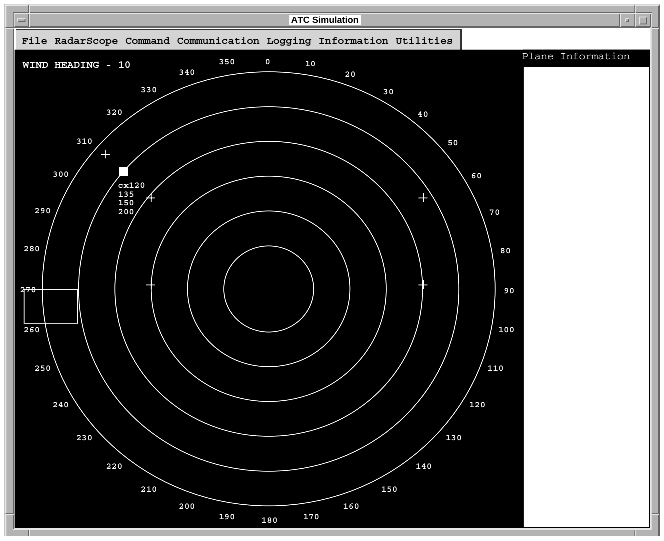

**Figure 4.** Snapshot of the ATC display.

block[5]. The track symbol is a solid white square. The data block shows the aircraft identifier, its heading ($135^{o}$), its speed (150 knots), and its altitude (20,000 feet, displayed in hundreds of feet). Fixes E, D, A, B, and Q are shown as white crosses. The wind heading in degrees appears in the upper left corner. Range rings and heading markers are also displayed.

The user interface is implemented in a Motif style (Open Software Foundation, 1990). This is important because it constrains the user interface widgets to conform to a certain style. This situation helps to simplify the task because classes of widgets can be recognised from their common features.

Control of the simulation is currently achieved by using menus and dialogue boxes. This makes the task simpler than real ATC but defines a starting point for modelling mouse-driven behaviour.

## Perception

Our implementation of perception has to be able to perceive objects that appear as part of the interface in a form that the cognitive model can use. The simulated eye has to be able to perceive, to locate aircraft, and to identify any data values, labels, and widgets that are displayed.

During a fixation[6], the simulated eye perceives data as numbers, symbols, and widgets (e.g., radio buttons) as opposed to images. Although we avoid the low-level signal processing issues, we still attempt to model relevant human perceptual limits. We have therefore implemented the fovea of the simulated eye as a rectangle, approximately the size of a thumbnail when held at arm's length. This roughly corresponds to the area of visual field that is projected onto the fovea in a real eye. Although we are working at a relatively high level of abstraction, the movement of the eye is still driven by the cognitive model. The model sends a command to the eye telling it to saccade. The fovea, shown as a white outline rectangle in Figure 4, appears at the place where the simulated eye is looking.

We have also incorporated a simplistic model of peripheral vision: anything on the display screen that is located outside the foveal rectangle is represented by its coordinate position. This information is used to allow the eye to perform a fixation reflex to locate an aircraft when it first appears.

## Motor action

To issue clearances, the simulated hand must use a mouse (currently the only way of interacting with the simulation). As with perception, the hand resides outside of the cognitive model, but is still driven by it. We have implemented a simplified model of the hand: it is represented by a mouse pointer which delivers a similar result to that which would be obtained from sending motor commands to drive a real hand to move the real mouse. We assume that there is no latency in locating the mouse and that movement of the mouse is governed by where the simulated eye is fixating (although we plan to improve on this simplification). The cognitive model controls the simulated hand by commanding it to perform mouse actions (e.g., press, release). The results of performing the actions are then detected by the simulated eye.

## Cognitive model

We have identified a number of requirements that models should meet. Our model satisfies several of these. It has some HCI knowledge, general problem solving skills, and ATC domain knowledge. It also incorporates the ability to learn. We have designed the model to handle incomplete domain knowledge and it can learn through post mortem reflection.

The model cognitively controls the simulated eye by means of saccades and fixations. At first the model fixates and receives from perception all field labels, data values, and widgets within the fovea plus location information for objects outside of the fovea. The model processes the input from the current fixation and saccades to one of the locations identified in the periphery when information from other places on the display is required.

The cognitive model also controls the hand. Our model tells the hand, represented as a mouse pointer, to move to various parts of

[5]The track symbol is displayed at the aircraft's position (in latitude and longitude). A data block is a set of fields associated with an aircraft that moves with the track symbol.
[6]A fixation is the period between eye movements when the eye remains (more or less) still so that objects are in clear focus.

the display. It also instructs the hand to press and to release the mouse button.

For now, the model has very limited HCI knowledge. For example, it knows it can issue clearances by using the menus and dialogue boxes, but all of the information concerning how to pull down the appropriate menu and to use the appropriate dialogue box is pre-defined. We are developing a model that learns what is on menus based on Ayn (Howes, 1994).

To add to the cognitive resources available to our model, we have implemented a generic problem solving capability: a model of generic deduction (based on Larkin, 1981 and Ritter, 1992). Our version of the deduction mechanism can reason using a causal chain with either quantitative or qualitative variables. We have also incorporated a simple model of memory and recall (Rosenbloom & Lee, 1989). These additions were relatively easy to make because the models we integrated were already implemented using Soar.

Our model learns during problem solving. However, not all methods of learning are applicable during problem solving. We have taken some account of this by including a theory of post mortem explanation-based reflection (Bass, 1995). This allows the model to assess the outcome of earlier decisions after doing the task.

### Example scenarios

When a landing aircraft enters the airspace and the model has noticed it, the model decides which arrival route (and hence which runway) to assign to this aircraft. The model initially knows to choose a runway so that the landing direction is into the wind. However, when the model tries to select a runway, it does not know which one to select because it does not know which way the wind is blowing and it has not considered the runway heading. Using its simulated eye, the model finds the wind heading. The model then uses deductive reasoning to determine which runway allows the aircraft to land into the wind. Its reasoning is as follows:

(1) Determine the wind compass direction based on the current wind heading.

(2) Determine the desired "into the wind" compass direction based on the wind compass direction.

(3) Determine the desired runway compass direction based on the "into the wind" compass direction.

The model learns from this experience and can directly apply its learned knowledge with subsequent aircraft.

Once the model selects a runway for a landing aircraft, it monitors the aircraft along its flight and issues clearances as appropriate. This is done by controlling the hand, which enters the clearances by using the menu bar and dialogue boxes.

During task performance, the model learns through its interaction with the simulation. For example, if an aircraft enters the airspace at Fix E when the wind direction is 10˚, the model learns to choose Arrival Route 1 so the aircraft can land into the wind. If the aircraft does not have enough fuel to fly this route, it crashes after flying over fix B, because the model does not know to consider fuel level during its initial problem solving.

After the crash, and only then, the model can learn through reflection (Bass, 1995). The model remembers the aircraft's initial state and the initial wind heading. It considers the decisions it has made and which features may have been important (by using deductive reasoning). It updates its domain knowledge to consider the aircraft's initial fuel level during problem solving. This shows how we integrate learning during problem solving with learning after problem solving, and serves to point out their mutual constraints and interdependence.

## Discussion

We have developed a cognitive model that can control a simulation of a simple ATC task. The model utilises simulated perception and motor action to interact with the simulation in psychologically plausible ways. In addition to performing the task, the model learns during and after problem solving.

The model is designed to be able to learn additional domain knowledge when situations for which it has incomplete or incorrect knowledge arise. It incorporates a new general theory of post problem-solving explanation-based reflection that allows it to learn from its interaction with the simulation. Our results suggest that less experienced operators may ignore aspects of the environment which are, in fact, an essential element of successful task performance. In our task, the fuel level of the aircraft turns out to be critical. Untrained operators may ignore this information if it is not currently displayed, or if they simply do not appreciate its importance.

The model is not yet a complete competence model in this area. If the simulation is initialised with the aircraft flying at high speed, aircraft can pass Fix D before the model has decided which runway to use. The model does not issue a heading change clearance at D so the aircraft flies away from the desired path. Although this scenario was an artificially created problem used to investigate how the model responds, it illustrates that the model is incomplete. In more realistic situations, where timing is critical and the task cannot be completed in a timely manner with a proposed interface, we would expect models to highlight such issues. However, to deal with this particular problem, we would want the model to have a default procedure (which runway to use) to follow. The issue then becomes one of the model deciding to use a different strategy because it does not have sufficient time to be able to use its analytic skills. This result points back to the requirement for having the ability to change strategies.

Creating this model highlighted what aspects of cognition and interaction are critical in performing the task. In considering extending the model to control more than one aircraft, it is clear that models of interaction must incorporate a theory of visual attention so the model can switch between aircraft. This attention cannot be a simple pointer, which would allow attention to switch repeatedly between aircraft; it must be influenced by planning and by expectations.

In implementing visual search, we found that extrafoveal vision is crucial. Without peripheral vision the model is reduced to a brute force scan to find things. Simple peripheral references are not enough though, because the model must remember which items in the periphery it has seen, and some type of episodic memory appears to be necessary as well.

We now know that perception of alignment in interfaces is important and such knowledge should be included. The model should realize that all the fields in one aircraft's data block belong together. As an aircraft moves on the display, its data block fields, all left-justified, move together at the same rate. Based on alignment and motion, the model should know (or should learn) which fields belong to which aircraft when multiple aircraft are near each other.

In fact, we have found that there is a whole range of visual effects that must be modelled. For example, spatial reasoning is critical for estimating future position based on current heading and speed. This capability is useful in planning, and when the model needs to refer back to an aircraft that has moved out of the foveal region. The planning need stems from ensuring that aircraft do not crash at some point in the future. Also the model must do some clearance conformance checking. When a clearance is issued, the model should look at the aircraft at some point in the future to see that the aircraft (i.e., the pilot) has taken action to conform to the clearance.

Our model makes suggestions about how to support learning in this task For example, in order to expand our theory of post problem-solving explanation-based reflection, the model should be able to recreate the initial situation instead of recalling it. We could add flight strips to our system as air traffic controllers use flight strips with hand-marked updates as written histories for each aircraft. If a controller needs to consider an aircraft's initial situation, the flight strip provides a means for recreating it.

We have identified several candidate sub-models to include. Each supports different aspects of ATC and interface use. For example, Huffman (1994) built a Soar model that learns from taking instruction; Mertz (1995) built a model that learns from "reading" a properly represented instruction manual, and Weismeyer (1992) built a model of visual attention.
We can now see that merely providing a model will not be enough. In order to make our technique more widely available, we will have to provide a much better interface to Soar including a system to interpret the model's behaviour. Experienced Soar analysts would have little problem using and understanding our models, but others would need help in understanding the model behaviour.

What we have done to date has indicated that developing cognitive models to control simulations is a large undertaking and much work remains including making them routinely reusable. When our prototype is complete, the best we can hope for is a general Soar model that can control and learn from simulations with Garnet interfaces. At present, we have defined an approach and have begun to implement a model that meets its requirements. The model incorporates domain, HCI, and general knowledge. We have shown that this model can control a simulation using simple models of perception and motor action.

## ACKNOWLEDGMENTS

Support has been provided by the DRA, contract number 2024/004, and by the ESRC Centre for Research in Development, Instruction and Training. The authors thank Michael Albers, Alan Chappell, Lynn Gathercole, Josef Nerb, Jennifer Turns, and Andy Weiner for their comments.

## REFERENCES

Amalberti, R., & Deblon, F. (1992). Cognitive modeling of aircraft process control: A step towards an intelligent onboard assistance system. *International J. of Man-Machine Studies,* 36(6), 639-671.

Bass, E. J. (1995). Learning through reflection after problem-solving. In *Proceedings of the Soar XV Workshop.*

Cacciabue, P. C., Decortis, F., Drozdowicz, B., Masson, M., & Nordvik, P. (1992). COSIMO: A cognitive simulation model of human decision making and behaviour in accident management of complex plants. *IEEE Transactions on System, Man and Cybernetics,* 22(5), 1058-1074.

Cacciabue, P. C., & Hollnagel, E. (1993). Simulation of cognition: Applications. In J. M. Hoc, P. C. Cacciabue & E. Hollnagel (Eds.). *Expertise and Technology: Cognition and Human-Computer Interaction.* Hillsdale, NJ: LEA.

Card, S., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *CACM,* 23(7), 396-410.

Card, S., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: LEA.

Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *J. of Experimental Psychology*, 71, 381-391.

Freed, M. & Johnston, J. C. (1995). Simulating human cognition in the domain of air traffic control. In M. T. Cox & M. Freed (Eds.), *1995 AAAI Symposium on Representing Mental States and Mechanisms.* Menlo Park, CA: AAAI Press.

Glass, G. (1993). *UNIX for Programmers and Users: A Complete Guide.* Englewood Cliffs, NJ: Prentice-Hall.

Hammond, K. R., Stewart, T. R., Brehmer, B., & Steinman, D. O. (1975). Social judgment theory. In M. F. Kaplan & S. Schwartz (Eds.) *Human judgment and decision processes*. San Diego, CA: Academic Press.

Howes, A. (1994). A model of the acquisition of menu knowledge through exploration. In *Proceedings of CHI '94,* 445-451. New York: ACM Press.

Howes, A. & Young, R. M. (1991). Predicting the learnability of task-action mappings. In *Proceedings of CHI '91,* 113-118. New York: ACM Press.

Huffman, S. B. (1994). Instructable autonomous agents. Ph.D. thesis. Dept. of Electrical Engineering and Computer Science, U. of Michigan, CSE-TR-193-94

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1995). Predictive engineering models using the EPIC architecture for a high-performance task. In *Proceedings of CHI' 95,* 11-18. New York: ACM Press.

Knaeuper, A. & Morris, N. M. (1984). A model-based approach to online aiding and training in process control. In *Proceedings of the 1984 IEEE International Conference on Systems, Man and Cybernetics,* 173-177.

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

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.) *Cognitive Skills and Their Acquisition*. Hillsdale, NJ: LEA.

Mertz, J. S. (1995). Using a cognitive architecture to design instructions. Ph.D. thesis, Carnegie Mellon University.

Mitchell, C. M. & Saisi, D. L. (1987). Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(4), 573-593.

Myers, B. A., Guise, 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. & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the power law of practice. In J. R. Anderson (Ed.) *Cognitive Skills and their Acquisition.* Hillsdale, NJ: LEA.

Ong, R. L. (1994). Mechanisms for routinely tying cognitive models to interactive simulations. M. Sc. thesis, U. of Nottingham. (ESRC Centre for Research in Development, Instruction and Training, TR #21.)

Open Software Foundation (1990). *Motif User's Guide.* Englewood Cliffs, NJ: Prentice-Hall.

Peck, V. A., & John, B. E. (1992). Browser-Soar: A computational model of a highly interactive task. In *Proceedings of CHI '92,* 165-172. New York: ACM Press.

Pearson, D. J., Huffman, S. B., Willis, M. B., Laird, J. E., & Jones, R. M. (1993). A symbolic solution to intelligent real-time control. *Robotics and Autonomous Systems*, 11, 279-291.

Rasmussen, J. (1985). The role of hierarchical knowledge representation in decision making and system management. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-15, 234-243.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction.* New York: Elsevier.

Reder, L. M., & Ritter, F. E. (1992). What determines initial feeling of knowing? Familiarity with question terms, not the answer. *J. of Experimental Psychology: Learning, Memory, and Cognition*, 18(3), 435-451.

Rieman, J., Lewis, C., Young, R. M., & Polson, P. G. (1994). "Why is a raven like a writing desk?" Lessons in interface consistency and analogical reasoning from two cognitive architectures. In *Proceedings of CHI '94,* 438-444. New York: ACM Press.

Ritter, F. E. (1992) Bruno Levy's Able-Soar, Jr. model. In *Proceedings of the Soar X Workshop*.

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

Rosenbloom, P. S. & Lee, S. (1989). Data-chunking memory package. The Soar Project. Information Science Institute, USC. Unpublished computer program.

Sasou, K., Takano, K., Yoshimuraa, S., Haraokab, K., & Kitamurac, M. (1995). Modeling and simulation of operator team behavior. In Y. Anzai, K. Ogawa, & H. Mori (Eds.) *Symbiosis of Human and Artifact.* New York: Elsevier Science.

Suchman, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge: Cambridge University Press.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. (1995). Intelligent Agents for Interactive Simulation Environments. *AI Magazine*, 16(1), 15-39.

Wade, N.J. & Swanston, M. (1991) *An Introduction to Visual Perception*. London: Routledge.

Weismeyer, M. D. (1992). An operator-based model of human covert visual attention. Ph.D. thesis, U. of Michigan.

Young, R. M., & Whittington, J. E. (1990). Using a knowledge analysis to predict conceptual errors in text-editor usage. In *Proceedings of CHI '90,* 91-97. New York: ACM Press.