Embodied Models as Simulated Users: Introduction to this Special Issue on Using Cognitive Models to Improve Interface Design

Frank E. Ritter[a] and Richard M. Young[b]

[a] School of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16801, USA.
ritter@ist.psu.edu

[b] Psychology Department, University of Hertfordshire, Hatfield, Herts AL10 9AB, UK.
r.m.young@herts.ac.uk

Running title: `Embodied Models as Simulated Users`

Contact author: Frank Ritter <`ritter@ist.psu.edu`>

Full contact address:

Frank Ritter

School of Information Sciences and Technology

Penn State

512 Rider Building

120 S. Burrowes St.

University Park, PA 16801

+1 (814) 865-4453 (direct line)

+1 (814) 865-5604 (fax)

Abstract

Cognitive models provide a means for applying what is known about psychology to the design of interfaces, thereby improving their quality and usability. Existing uses of models include predicting time and errors for users to perform tasks, acting as embedded assistants to help users perform their tasks, and serving as surrogate users. Treating the design of human-computer interfaces as a form of engineering design requires the development and application of user models. A recent trend is for models to be built within the fixed framework of a cognitive architecture, which has been extended by the addition of simulated eyes and hands, enabling the construction of embodied models. Being embodied allows models to interact directly with interfaces. The resulting models can be used to evaluate the interfaces they use, and serve as explanations of users' behavior. The papers in this Special Issue point to a new route for the future, one in which models built within embodied cognitive architectures provide information for the design of better interfaces.

1. Cognitive Models in HCI

Computational cognitive models are simulation models realized as computer programs that mimic the performance of human cognitive skills. They can exhibit a wide variety of behavior, ranging from reactive behavior (Jones, Laird, Nielsen, Coulter, Kenny, & Koss, 1999; Sloman & Logan, 1999) to deliberative problem solving (Lohse, 1997; Peck & John, 1992; Sloman & Logan, 1999). There are now cognitive models that include emotion and other behavioral moderators (Belavkin, Ritter, & Elliman, 1999; Jones, 1998; Jongman, 1998; Ritter & Avraamides, 2000), and others that simulate developmental processes (Jones, Ritter, & Wood, 2000) and predict errors (Gray, 2000). Cognitive models are increasingly able to interact with an independent simulation of their external environment. It is now possible to study how interactive behavior emerges out of the constraints and opportunities provided by the interaction of embodied cognition with the task being performed and with the artifact (interface or device) provided to support task performance. This capability for interaction opens up the possibility of applying models to human-computer interaction (HCI) in a more direct way than has been feasible so far.

In this introduction to the Special Issue on using cognitive models to improve interface design, we first examine the current use of models in interface design and review recent trends. We then outline a newly emerging role for models and indicate how each of the papers in the Special Issue contributes to this vision.

1.1 Main ways in which models have been applied

John (1998) lists three main ways in which cognitive models have been applied to HCI: to predict time and errors, to assist users, and to act as surrogate users.

The first way to apply cognitive models in HCI is to use them to predict relevant aspects of human performance, such as time and errors (Howes, 1995; Sears, 1993). The Keystroke Level Model, and particularly the GOMS family of models (Card, Moran, & Newell, 1983; John & Kieras, 1996) have been deployed successfully in the laboratory and in industry. These are descriptive models that predict the time to perform tasks with an interface, perhaps even before the interfaces are fully built and committed to. They themselves do not indicate how the information processing required to perform the task will be done. These analytical models can help create and choose better designs, sometimes saving millions of dollars (e.g., Gray, John, & Atwood, 1993). New tools are emerging to help make the application of such models more tractable, and hence allow them to be used more routinely (Beard, Smith, & Denelsbeck, 1996; Williams, Severinghaus, & Clare, 1998).

The second way is to use cognitive models as embedded assistants to help human users. In particular, computational process models — that is, those that simulate how users perform a task by actually performing it themselves — can be used to guide the interaction in order to help users with their tasks. A practically important application of such models is as "cognitive tutors" in school education (Anderson, Corbett, Koedinger, & Pelletier, 1995). These models know what task the student is attempting, and can provide specialized support based on that knowledge.

The third way is for models to substitute for users. Computational process models specify the information needed to perform a user's task, the way that information is processed, and the steps taken to perform the task in detail. These models are functional in the sense that they can themselves perform the task and can show how different designs lead to different behaviors, what knowledge should be included in manuals, and why users have trouble with particular activities. This type of cognitive model has been used to populate synthetic environments (Pew & Mavor, 1998), for example to simulate fighter aircraft crews (Jones et al., 1999). Other researchers have proposed employing models as simulated users to test interfaces (Byrne, Wood, Sukaviriya, Foley, & Kieras, 1994; Gray & Fu, 2001; Lohse, 1997; St. Amant, 2000), but the technique has not been widely adopted.

1.2 Recent developments supporting models as users

The possibility of employing cognitive models to test the usability of interfaces has a relatively long history (Card et al., 1983; Kieras, Wood, Abotel, & Hornof, 1995a; Kieras, Wood, & Meyer, 1995b; Olson & Olson, 1990; Wherry, 1976). One of the most important features of the idea is that user models can be used to make quantitative predictions about user behavior and comparative predictions between interfaces.

Descriptive cognitive models have been developed for making predictions about user performance (such as time and actions to perform a task) at the computer interface. Many of these models are reviewed by Howes (1995). For example, Task Action Grammar (Payne & Green, 1986) predicts the relative speed of learning variants of an interface, and GOMS models can be used to predict how long it will take an expert to perform a particular task. Typically, however, such models have not been developed as process models under the constraints of an explicit cognitive architecture.

Computational process models, such as models of menu exploration (Rieman, Young, & Howes, 1996), browsing (Peck & John, 1992), and analogical reasoning (Rieman, Lewis, Young, & Polson, 1994) interact with their own interface simulations, which while interesting in their own right, do not provide any support for applying the models to additional interfaces. These and other examples have generally been small efforts and have had limited impact because they are not yet easy enough to use, do not incorporate sufficiently comprehensive theories, and do not work with already existing interfaces or designs in progress.

Two developments of the last decade have radically altered the nature of the models, the way in which cognitive modeling is done, and the models' potential applicability to HCI. One development is the emergence of cognitive architectures, the other is the addition of simulated "eyes and hands", which creates an embodied or integrated (Pew & Mavor, 1998) cognitive architecture.

1.2.1 Cognitive architectures

A recent trend in work extending the state of the art on modeling users has been to cast the models within a *cognitive architecture*. A cognitive architecture embodies a scientific hypothesis about those aspects of human cognition that are relatively constant over time and relatively independent of task. Examples range from architectures claiming broad scope, such as Soar (Newell, 1990) and ACT-R (Anderson & Lebiere, 1998), to more specialized ones such as C-I (Construction-Integration: Kintsch, 1998). The alternative to working with an architecture is for the analyst to implement a user model directly in some programming language, such as Lisp or C, chosen for its ease of programming and its convenience for expressing the model, but not for its theoretical contribution.

In models built within a cognitive architecture, part of the content of the model is supplied by the architecture itself, while the rest is supplied by what the analyst has to add to the (generic) architecture in order to construct a (specific) model. All models constructed within a given cognitive architecture share the features of that architecture. What distinguishes one model from another is the extra information that has to be provided in order to define a particular model within the architecture. The theoretical content of the model is distributed between the cognitive architecture and the knowledge added to specify the model (Howes & Young, 1997).

Constructing a user model within a cognitive architecture offers various advantages to the modeller. The cognitive architecture constrains, or biases, the kinds of models that can be constructed within it. One of the consequences is that the constrained models serve to identify what knowledge is needed in order to perform the tasks being analyzed, what information is drawn from the environment, and what knowledge is inherent to the problem solver.

Furthermore, because all the models built within a given cognitive architecture share certain assumptions (namely, those embedded in the architecture), it should be easier to integrate separately-developed models of narrow scope into a coherent, single model of broader scope. Over time, this should allow the development of libraries of idioms and partial models, which in turn will ease the task of constructing new models.

Another advantage is that cognitive architectures typically provide cognitive mechanisms and resources that are also relevant to interface use. Such measures can include: the working memory load when using the interface to accomplish a task; the time taken to learn an interface; what gets learned; and the causes and types of errors in using the interface.

Finally, cognitive architectures provide a basis for teaching designers about users. Cognitive architectures provide a coherent theory of what generates human behavior. This theory will support designers' understanding of users as components of interactive systems in terms similar to other components of the systems (Barnard, May, Duke, & Duce, 2000; Duke, Barnard, Duce, & May, 1998). Designers can refer to the architecture and its behavior for answers to both general and specific questions about users.

1.2.2 Embodied cognitive models

Another trend in creating cognitive models is to embody them, that is, to give them a way to interact with a real or simulated world. Such embodiments have typically provided models with simulated eyes and hands (Byrne & Anderson, 1998; Kieras & Meyer, 1997; Ritter, Baxter, Jones, & Young, 2000). Other modalities including hearing and touch are sometimes included (e.g., Salvucci's model, this issue) and will need to be added more routinely in the future. This approach contrasts with earlier work where either simple tasks were represented entirely within the model, or inputs were provided in pre-processed form and outputs were treated as atomic and errorless no matter what their complexity.

Providing models with eyes and hands helps the models behave more like users with respect to timing and with regard to the problems of interacting with an interface. Having eyes and hands forces the models to interact in order to extract information from their environment, and they have to interact and sometimes solve problems to implement their actions. Additional knowledge is required, and new behavior emerges because not all information on the display is immediately available, and not all steps can be carried out as simple actions. Thus the models are situated in an environment because not all of their behavior is internal. Much of their behavior arises from interaction such as finding information in a visual scene or targeting the mouse.

There are several advantages to providing cognitive models with simulated eyes and hands. It leads to the creation of better psychological models because the models are more complete and have access to a richer world. Because these embodied models are more complete they can be more widely applied as substitutes for users in studies, simulations, and games. Such models may also be better for helping users because they more accurately reflect what users are doing and what their current goals are, including those concerning interaction. Finally, these embodied models can be more readily applied to predict the use and usability of interfaces because it becomes easier to put the model in contact with an interface to be evaluated. The papers in this special issue demonstrate most of these advantages.

2. An Engineering Approach to HCI Design

We believe that HCI design should be viewed as a branch of engineering design (Dowell & Long, 1989; John & Kieras, 1996). In their seminal work, *The Psychology of Human-Computer Interaction*, Card, Moran and Newell (1983, p. 11-12) identified design as "where the action is in the human-computer interface". Moreover, they proposed that "...the primary professionals—the computer system designers—be the main agents to apply psychology". This is merely following the history of good engineering practice in other disciplines where, for example, chemical engineers have come to understand the need to apply the relevant chemistry in the building of process plants and electrical engineers apply the relevant electronics knowledge. The natural corollary is that user interface designers have to understand and apply the relevant psychology in designing human-computer interfaces. Doing this by hand can be difficult. In order for it to be done routinely, the designer must have tools and support for exploring design trade-offs easily.

2.1 Design tools in engineering

In many areas, designers routinely create models to understand trade-offs better. Because design is essentially an iterative process, these models can then be tested and the resultant feedback used to modify the design. This use of models is as true of Civil Engineering, where scaled physical models of structures may be tested in wind tunnels, as it is of Electronic Engineering, where the model is based more on information than on materials and is often computer-based. When much knowledge is needed or many details need to be tracked, automated tools are nearly always used.

In some design domains there are already a number of automated design support tools in existence. For example, the age of heuristic evaluation of electrical circuits has long passed. Tools like SPICE (Thorpe, 1992) provide the electrical engineer with a tool to describe a circuit and include a theory describing how that circuit will behave given a set of signal inputs. Although SPICE does not directly give hints on how to improve the circuit, it does provide displays of the circuit's performance. Depending on the version of SPICE, other outputs may also be generated, for example: components used, possible timing problems, and violations of design constraints such as minimal distance between parts. In the hands of a designer, it can be used to identify the parts of the circuit that are likely to fail, confirm that the circuit will perform the required task, and assist in understanding the circuit's behavior. Rarely does a circuit need to be physically built and tested in order to see if and how it works.

In order to simulate a circuit, SPICE requires several things: a full specification of the circuit, a description of the components that will be used in the circuit, and a specification of any input wave forms the circuit will see. It includes a unifying theory of how the components interact with each other. We believe that we can start to map some of this approach across to interface design by using cognitive architectures to realize the components of the user model and provide a theory to tie them together.

2.2 Existing HCI design tools

While the ability to evaluate user interfaces routinely does not yet exist, a few tools have been developed to facilitate the assessment of interfaces with user models as part of an iterative process of interface design. There are two areas that provide lessons for our work: tools that examine existing interfaces, and tools that include cognitive models.

2.2.1 Tools that directly analyze interfaces

Shneiderman and his colleagues (Mahajan & Shneiderman, 1995) have developed a set of tools to analyze and evaluate interfaces implemented in Visual Basic. These tools examine the Visual Basic code and detect inconsistencies in the interface, the use of synonyms, variant capitalization, and misalignment of graphical items. The designer then can use these reported errors to improve the interface. Working directly with a popular prototyping language allows these tools to be applied quite widely. The tools perform only a static analysis of the interface, however, and are currently limited to testing relatively simple properties. Indeed, they can be regarded as implementing the most basic of style guidelines.

A model-based approach to evaluating user interface design has already been employed in a number of tools that focus mainly on interface development, such as HUMANOID (Szekely, Luo, & Neches, 1992, 1993). Other design tools have included support for static evaluation of the resulting interfaces. For example, USAGE (Byrne et al., 1994) incorporates a formal analysis technique to evaluate interfaces created in an interface design environment.

2.2.2 Tools that include cognitive models

Some existing tools employ user models to evaluate interfaces. These have included declarative models, such as the keystroke model (Nichols & Ritter, 1995) and GOMS (Beard et al., 1996; Williams et al., 1998), that make predictions about total time given a description of the user's behavior; performance models that predict what aspects of cognition are used in performing a task (May, Barnard, & Blandford, 1993); and process models that, by actually performing the task, predict behavior, time, and resource usage (Huguenard, Lerch, Junker, Patz, & Kass, 1997; Kieras et al., 1995b). The value of employing architecturally-based cognitive models to help the design of the interface has been demonstrated with the cognitive simulation model COSIMO (Cacciabue, Decortis, Drozdowicz, Masson, & Nordvik, 1992). COSIMO utilizes a blackboard architecture to provide the required level of flexibility and modularity, and to allow it to model the various types of cognitive functions, such as information seeking and monitoring, performed by operators of a nuclear power plant simulation.

APEX (Freed & Remington, 1998; Freed & Remington, 2000) is a tool for modeling and simulating human operators in task environments that demand reactive, multitasking behavior. It is intended for use at an early stage in the process of designing interfaces and procedures where its ability to identify certain usability problems can have the greatest benefit. It has been applied so far to cockpits and air traffic control systems. It includes a model of high-level cognition for deciding and controlling action that is part of an embodied architecture. The interfaces it can interact with are built with a special UIMS. Users cannot interact with the same interface, although analysts can watch the model interact with the interface. APEX has moved some way towards being able to model the effects of interaction, showing how visual displays can support problem solving and how errors can arise in air traffic control.

Perhaps the most significant and current effort in automatically applying user models to interfaces is GLEAN (Kieras et al., 1995a). It is an environment consonant with our view that designers should be able to automatically and routinely evaluate interface designs. In this tool, the designer inputs a GOMS description of the user's tasks, a set of benchmark tasks, and an abstract representation of the interface. GLEAN then generates usability metrics in two ways. The static GOMS descriptions provide measures of learning time and interface consistency. Simulating the user-device interaction provides usability metrics such as execution time.

GLEAN does several things well. It appears to make GOMS models — an already robust approach — easier to use, and it automates their execution. It can also amortize the creation and use of the models by providing some of them in libraries. Finally, it provides multiple types of information to designers.

There are several ways GLEAN could be improved. The user model does not directly interact with the interface being developed. Currently, the designer must create an additional representation of the interface for the model. The need for this separate, abstract copy makes it harder to develop variations of the model that use different aspects of the interface, for they would require modifications to the abstracted interface as well. It also does not yet support the modeling of perceptual features (and perhaps certain forms of visual search), of how learning occurs, and of how errors might arise from lack of knowledge or problems with perception and action. It does, however, partially illustrate what such a tool would look like.

2.3 The future: Model-Based Evaluation of Interfaces

The recent availability of embodied cognitive models (i.e., cognitive models equipped with simulated eyes and hands, as discussed in Section 1.2.2) opens the way to a fundamentally different approach for applying models to interface evaluation. Figure 1 shows the relationship between an embodied model and the interface that the user sees. Although the model does not physically use the input devices of mouse and keyboard or optically see the display, so far as the running software is concerned it does use the same interface: it has access to the information on the display, and it generates events that are interpreted as key presses and mouse clicks.

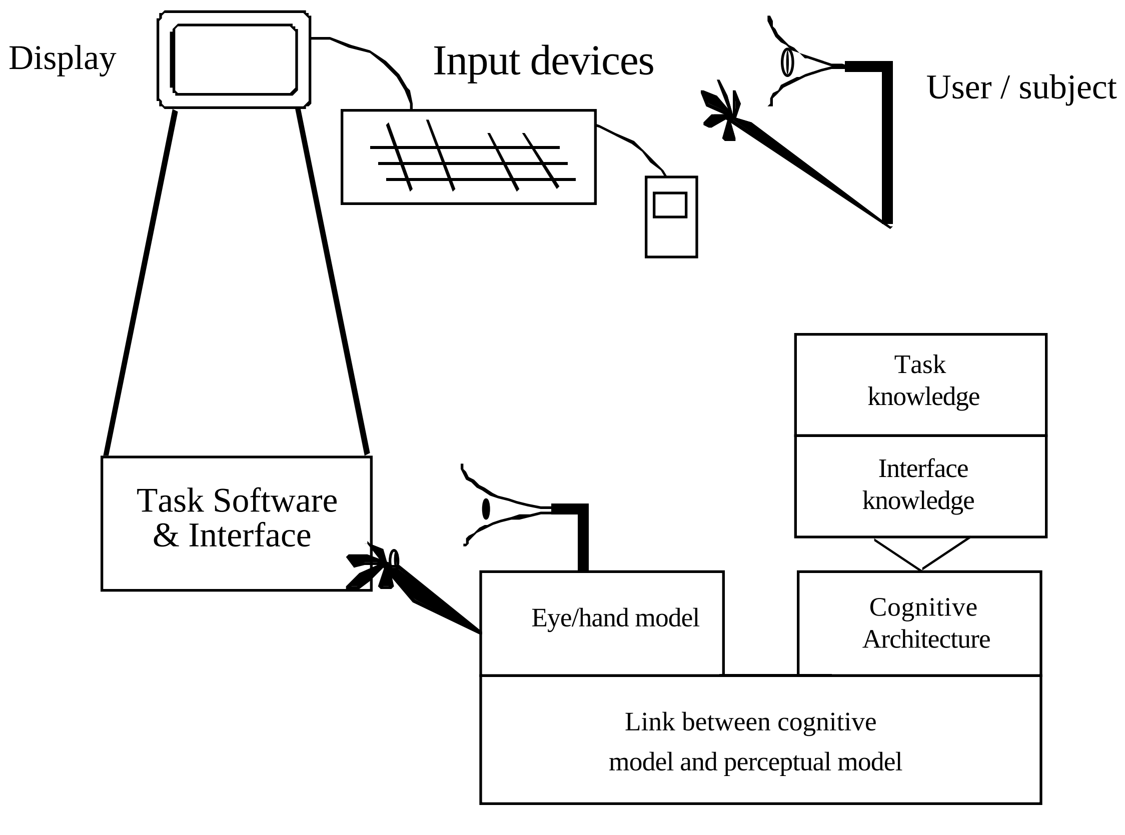

Figure 1. Components necessary to creating a system to provide an automatic user. Not shown is an environment to create and manage user model libraries.

Unlike a descriptive model, an embodied cognitive user model applied in this way itself performs the task. In consequence, it has unique contributions to make to the evaluation of an interface. First, embodied models can provide automatic evaluation of aspects of the interface that depend upon the user's dynamic performance, rather than being restricted to static measures. This broadens the range of aspects of the interface that can be checked. Second, because an embodied user model interacts directly with the interface and produces an unfolding performance comparable to that of the user, this kind of model can serve to put designers into contact with the user's situation — can help locate designers in the user's shoes — better than descriptive, less dynamic tools for evaluation that designers have to apply by hand. And third, this form of evaluation can be carried out directly with an early version of the interface, without requiring the construction of a simulation or other redescription of the interface.

2.3.1 Relationship to user testing

Initially, the application of user models to evaluate interface designs will complement user testing as currently performed. Simple and common problems will be found by the models, and more complex problems in more complex parts of the interface will continue to have to be found by user studies. But in the medium to longer term, one of the advantages of this approach is the reduced dependence on gathering user data. In other areas of engineering, the use of simulations has drastically decreased the need for physical models in design.

Using models to test interfaces may currently be more expensive than running subjects, but when the approach is fully implemented the situation will reverse. Running users is inherently expensive, and suffers several other disadvantages. Subjects new to an interface can use it only for a short period before they are no longer new to it. These subjects cannot be re-run on a variant interface for comparison because of the interference from their previous learning. There can be large differences between subjects that are not apparent a priori. But most of all, data from users can tell the designer only *what* happens with a given interface; a model can also tell *why* it happens, providing directly useful feedback to the designer (Young, Green, & Simon, 1989). This ability of models, unlike usability tests, to explain results makes them applicable to the interactive behavior that emerges from a wide range of tasks and interfaces.

Finally, there is the question of time. Allowing the model to interact with the interface in the same way that a real user would, generates behavior that mimics real users. One particular advantage is that the model's behavior can be generated and recorded in simulated real-time. This contrasts with the gathering of usability data using video analysis, where not only is the data gathering resource-intensive, but the transcription of the collected data typically takes an order of magnitude longer than the data gathering itself.

The need for user testing will not disappear, but its role will change. Instead of being applied directly in the process of iterative design, it will be used mainly for the initial development and validation of models. Direct user testing will be confined mainly to the final acceptance of a new design.

2.3.2 Limitations of the approach

The approach being proposed here has limitations, even beyond any limitations in the models themselves. Some of these limitations can be resolved with further work, but others are inherent to the approach.

There is a need for designers to work with preliminary and sketchy designs, at an early stage of the design process. Some interface evaluation tools have supported work with outline designs, others have not. As we are interested in the dynamic behavior of interfaces, having models interact with sketchy designs requires the development of a qualitative theory of interaction, which will provide an additional set of challenges (Barnard et al., 2000). Initially we may have to accept an incomplete tool that supports only iterative design and testing with complete interfaces, as is true with SPICE, and rely on experience with the tool to improve the designer's ability to apply the theory earlier in design. Where development is based on existing software and is not done from scratch, the models will have a place to start, even before the new designs are complete.

A model-based interface evaluation tool will still require a knowledgeable designer. Designers will have to understand the theories their tool implements, which in this case are theories of the user as represented by cognitive architectures. To make use of the tool, the designer will also have to understand the design space and the trade-offs inherent in it. For example, even with an interface evaluation tool as a computational aid, choices between the ease of use of the interface by experts and the learnability of the interface by novices still have to be made by the designer. This need for a knowledgeable designer to make informed choices is inevitable, so long as interface design problems remain without agreed or best solutions to multiple criteria. In areas where fixed solutions can be offered, these solutions can be incorporated into the modeling tool as they are in other engineering design tools.

Using cognitive models to interact with interfaces focuses on user behavior at a detailed level of interaction. Modeling interaction is currently a rather restrictive and specialized corner of user modeling that will be appropriate mainly, for example, when the user's time is expensive, errors are critical, it is difficult to run a usability study, or there are many potential users. The approach does not currently support higher order effects that occur in group work (although, see Carley 1996), but these may be important in the future. As interface design becomes more routine and becomes more like other areas of design, designers will increasingly care about the details of interaction (Gray & Boehm-Davis, in press), just as other designers do when there is enough knowledge to evaluate designs and when all costs, including user's time and errors, are important and can be measured and adjusted.

3. Papers in this Special Issue

The papers included in this special issue illustrate several of the components necessary to move the approach forward. The first paper, by St. Amant and Riedl, shows how tying models to interfaces can be supported in a general way though the application of simple vision algorithms. When complete this approach will provide a way for models to interact with a wide range of interfaces. The second paper, by Byrne, presents a general theory of interaction that has been incorporated in the ACT-R cognitive architecture. It provides additional support for the results to be reused widely and provides a set of mechanisms and constraints from psychology. The final paper, by Salvucci, illustrates what the application of these models to interface design might look like. The paper investigates how two different car phone interfaces affect driving behavior.

3.1 Technical foundations for modeling interaction (St. Amant & Riedl)

In order for an embodied model to use the same interface as the user, it must be able to see the same things on the display as the user does. Early work achieved this aim by instrumenting an individual interface with function calls. At present, it can be done in a fairly general way by accessing the internal structures in the interface (Byrne, this issue; Ritter et al., 2000).

A more general and robust approach is to allow direct access to input and output devices, such as the display screen and the mouse. Doing so provides a way for a model to manipulate all interfaces in an interactive system. The biggest challenge lies in interpreting the bitmap as a collection of meaningful objects, such as scrollbars and buttons.

St Amant and Riedl's paper on ***"A perceptual/action substrate for cognitive modeling in HCI"*** shows how such a technique enables models to interact with the same interface as users. His work has been used to tie models and agents to interfaces, and is available (as noted in the paper) for reuse by others. It supports the aims of allowing models to cover a more complete range of human behavior, including perception and action, and of providing models with access to a richer world. It permits the possibility, for example, of tying ACT-R models to any interface that runs on a PC. Doing this will allow models routinely to use interfaces built by others.

3.2 Including interaction in a cognitive architecture (Byrne)

Once the model has access to objects on the display it also has to be constrained to see only as much as users do and to encounter the same limitations. The model also has to be tied closely to a cognitive architecture because perception, cognition, and motor behavior are closely entwined. Embedding the simulated eyes and hands in an architecture creates an embodied architecture.

Byrne's ***"ACT-R/PM and menu selection: Applying a cognitive architecture to HCI"*** describes ACT-R/PM, a set of perceptual and motor modules for the ACT-R cognitive architecture. These modules focus specifically on the "eyes", both input (vision) and motor (eye movements), but also include a hand to type and to move the mouse. They allow commands to be sent to the perceptual and motor modules and performed in parallel, and they incorporate some established constraints on performance.

The resulting behavior is not perfect, but each new version of the architecture covers more and more of the relevant human behavior. Byrne's paper provides an example application, and other applications have been and are being developed. Numerous models are starting to use these eyes and hands, and they featured prominently in a recent conference on cognitive models (Ritter, 2001). ACT-R/PM models have also been used to compare interfaces: our final paper provides an example.

3.3 Application of model user to compare interfaces (Salvucci)

Salvucci's paper on ***"Predicting the effects of in-car interface use on driver performance: An integrated model approach"*** provides an illustration of the approach being applied to an important real-world task. The model is set up to explore how different dual tasks and different interfaces for the secondary task modify the primary task to a greater or lesser degree. The model shows that simultaneously driving and using a car phone leads to degradation in both tasks. The model quantifies the effects, and shows how and why the effects of dialing on driving vary between carphone interfaces. The resulting prediction, that voice input car phones are safer in that they reduce driver variance, provides exactly the type of help we hope for from such models.

The resulting model is one of the best current models of driving behavior, particularly with regard to dual-tasking. This is partially due to the fact that the model can interact directly with a driving simulator and a car phone simulator that users can interact with as well. The resulting model could also be used to drive cars in the simulation to study group behavior when several drivers are dialing phones.

4. Conclusions

The papers in this special issue lay the foundation for, and illustrate, an approach using embodied cognitive architectures to provide a framework for integrating what is known about users in the service of interface design. The resulting user model can interact with the interface to help guide design. These papers include several state-of-the-art examples of models built within integrated cognitive architectures. They use simulated eyes and hands to interact directly with the interface being evaluated.

These papers also suggest places where further work is required. The models of interaction themselves will remain incomplete for some time. For example, the models of perception are incomplete in numerous ways. The current models have trouble seeing white space (where objects are not) and seeing emergent features. The models learn, but they could learn more. They perform several different tasks, but they could perform more tasks. The models, while they are becoming easier to apply and more general, could be even easier to apply and even more applicable. The reuse of models and the creation of libraries of user models remains an open area of research and application. Finally, an application environment to deliver the models to an HCI designer is necessary. Such an environment would allow the designer to specify the tasks that user model will perform, and help the designer interpret the model's behavior.

The route towards model-based evaluation of interfaces will not be short and even, but the work presented by the papers in this special issue shows that it has become more feasible. The collection shows that models can be tied directly to human-computer interfaces, and that when they are, the models can provide useful predictions of how users will behave. In time, the routine application of user models in HCI design will provide a way to create better interfaces.

Acknowledgments

Suggestions and comments from Mike Byrne, Michael Freed, Wayne Gray, and Dario Salvucci have significantly improved this Introduction. Gordon Baxter helped develop this work during numerous discussions. We also would like to thank the reviewers of all the papers submitted to this special issue.

Papers in this Special Issue

Byrne, M. D. (2001) ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, XXX*, 000-000.

St. Amant, R., & Riedl, M. O. (2001) A perceptual/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies, XXX*, 000-000.

Salvucci, D. D. (2001) Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies, XXX*, 000-000.

References

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.

Barnard, P. J., May, J., Duke, D., & Duce, D. (2000). Systems, interactions, and macrotheory. *ACM Transactions on Computer-Human Interaction, 7*, 222-262.

Beard, D. V., Smith, D. K., & Denelsbeck, K. M. (1996). Quick and dirty GOMS: A case study of computed tomography interpretation. *Human-Computer Interaction, 11*, 157-180.

Belavkin, R. V., Ritter, F. E., & Elliman, D. G. (1999). Towards including simple emotions in a cognitive architecture in order to fit children's behaviour better. In *Proceedings of the 1999 Conference of the Cognitive Science Society*. 784. Mahwah, NJ: Lawrence Erlbaum.

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.

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

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

Carley, K. M. (1996). A comparison of artificial and human organizations. *Journal of Economic Behavior & Organization, 31*, 175-191.

Dowell, J., & Long, J. (1989). Towards a conception for an engineering discipline of human factors. *Ergonomics, 32*(11), 1513-1535.

Duke, D. J., Barnard, P. J., Duce, D. A., & May, J. (1998). Syndetic modelling. *Human-Computer Interaction, 13*, 337-393.

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.

Freed, M., & Remington, R. (2000). Making human-machine system simulation a practical engineering tool: An APEX overview. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the 3rd International Conference on Cognitive Modelling*. 110-117. Veenendaal (NL): Universal Press.

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science, 24*(2), 205-248.

Gray, W. D., & Boehm-Davis, D. A. (in press). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*.

Gray, W. D., & Fu, W.-t. (2001). Ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head: Implications of rational analysis for interface design. *CHI Letters, 3*(1).

Howes, A. (1995). Cognitive modelling in HCI. In A. Monk & N. Gilbert (Eds.), *Perspectives on HCI: Diverse approaches*. 97-119. London, UK: Academic Press.

Howes, A., & Young, R. M. (1997). The role of cognitive architecture in modeling the user: Soar's learning mechanism. *Human-Computer Interaction, 12*, 311-343.

Huguenard, B. R., Lerch, F. J., Junker, B. W., Patz, R. J., & Kass, R. E. (1997). Working memory failure in phone-based interaction. *ACM Transactions on Computer Human Interaction, 4*(2), 67-102.

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.

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. (1998). Modeling pilot fatigue with a synthetic behavior model. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavioural Representation*. 349-357.

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.

Jongman, G. M. G. (1998). How to fatigue ACT-R? In *Proceedings of the Second European Conference on Cognitive Modelling*. 52-57. Nottingham: Nottingham University Press.

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.

Kieras, D. E., Wood, S. D., Abotel, K., & Hornof, A. (1995a). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95)*. 91-100. New York, NY: ACM.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1995b). Predictive engineering models using the EPIC architecture for a high-performance task. In *Proceedings of the CHI '95 Conference on Human Factors in Computing Systems*. 11-18. New York, NY: ACM.

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.

Mahajan, R., & Shneiderman, B. (1995). A family of user interface consistency checking tools (Tech. Report No. CAR-TR-770). Human-Computer Interaction Laboratory, U. of Maryland.

May, J., Barnard, P. J., & Blandford, A. (1993). Using structural descriptions of interfaces to automate the modelling of user cognition. *User Modelling and Adaptive User Interfaces, 3*, 27-64.

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

Nichols, S., & Ritter, F. E. (1995). A theoretically motivated tool for automatically generating command aliases. In *CHI '95, Human Factors in Computer Systems*. 393-400. New York, NY: ACM.

Olson, J. R., & Olson, G. M. (1990). The growth of cognitive modeling in human-computer interaction since GOMS. *Human Computer Interaction, 5*(2 &3), 221-265.

Payne, S. J., & Green, T. R. G. (1986). Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction, 2*, 93-133.

Peck, V. A., & John, B. E. (1992). Browser-Soar: A computational model of a highly interactive task. In *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems*. 165-172. New York, NY: ACM.

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.

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 the CHI '94 Conference on Human Factors in Computing Systems*. 438-444. New York, NY: ACM.

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. (2001). Review of the Third International Cognitive Modelling Conference. *Cognitive Systems Research, 1*(4), 251-252.

Ritter, F. E., & Avraamides, M. N. (2000). Steps towards including behavioural moderators in human performance models in synthetic environments (Tech. Report No. 2000-1). Applied Cognitive Science Lab, School of Information Sciences and Technology, Penn State. http://ritter.ist.psu.edu/papers/steps-beh-mod.2.pdf.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 1-33.

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

Sloman, A., & Logan, B. (1999). Building cognitively rich agents using the Sim_Agent toolkit. *Communications of the Association of Computing Machinery, 42*(3), 71-77.

St. Amant, R. (2000). Interface agents as surrogate users. *intelligence* (Summer 2000), 29-38.

Thorpe, T. W. (1992). *Computerized circuit analysis with SPICE: A complete guide to SPICE, with applications*. New York, NY: Wiley.

Wherry, R. J. (1976). The Human Operator Simulator - HOS. In T. B. Sheridan & G. Johannsen (Eds.), *Monitoring behavior and supervisory control.* 283-293. New York, NY: Plenum Press.

Williams, K., Severinghaus, R., & Clare, T. (1998). A modified GOMS cognitive task analysis technique for creating computational models of adversary behavior. In *Proceedings of the Seventh Conference on Computer Generated Forces and Behavioral Representation*. 75-86.

Young, R. M., Green, T. R. G., & Simon, T. (1989). Programmable user models for predictive evaluation of interface designs. In *CHI'89 Conference Proceedings: Human Factors in Computing Systems*. 15-19. New York, NY: ACM Press.