Ritter, F. E. (2019). Modeling human cognitive behavior for system design. In S. Scataglini and G. Paul (Eds.), DHN and posturography, (pp. 517-525, Ch. 37). London: Academic Press.

Modeling Human Cognitive Behavior for System Design

Frank E. Ritter (frank.ritter @psu.edu)
College of IST
Penn State
University Park, PA 16802

27feb19
cognitive-models-in-designV6.doc

Abstract

The use of models of human cognitive to help in system design has been a targeted use-case nearly since these models were created. These models can be represented as a combination of fixed mechanisms (the architecture) and task knowledge (the learned subtasks) that uses these mechanisms to generate behavior. I first describe how these models might be used and their types of models and architectures and their uses, and then conclude with what appears to be necessary to use these models more routinely.

Acknowledgements

This was supported by ONR projects (N00014-18-C-7015, N00014-15-1-2275) and informed by a project with Harris Corporation on interface design. Comments from Cesar Colchado, Jacob Oury, Raphael Rodriguez, Robert St. Amant, Trevor Sands, Farnaz Tehranchi, and Shan Wang helped improve this paper.

1. Introduction

Models of human cognition can be represented as a combination of fixed mechanisms that constitute the architecture and a body of task knowledge that uses these mechanisms to generate behavior. There are a wide range of models and mechanisms that can be partially ordered by how formal they are. These models have been created to predict human behavior as a type of scientific theory.

Marvin Minsky at one point noted that if you were going to think about thinking, you were going to have to think about thinking about *something*. Given the roots of cognitive models in computer and organizational science, a natural thing for these models to think about would be how to use a computer interface, and thus to help in system design (e.g., Card, Moran, & Newell, 1982) . In addition, the creators of these models, nearly since the models were first created, have envisioned their use in system design (e.g., Byrne & Gray, 2003; Kieras, 1985; Ritter, 1993). In addition to the work cited here, Pew (2008) provides a review of some of these early models, and there has also been a series of government reports encouraging the use of cognitive models in system design (Elkind, Card, Hochberg, & Huey, 1989, 1990; Pew & Mavor, 1998, 2007).

Summary and overview of this chapter

In this brief chapter I first describe how these models might be used in system design, then categorize the types of models and their uses from the fields of human-computer interaction and cognitive science. I conclude by describing what appears to be necessary to use these models more routinely. I will use the term cognitive model both because it is shorter and because the literature I draw on prefers this term to modeling human cognitive behavior.

Limitations

This review is not comprehensive; it does not include all the models that have been used. There are further reviews that can provide a wider review of models and their types (Pew & Mavor, 1998, 2007; Ritter et al., 2003). There are ongoing communities of research in this area not covered in this review but that should be mentioned to guide the reader interested in knowing more. For example, the digital human bodies that are already used in design (*cite to chapters in this book*) are not reviewed here. Other communities include social simulation examining things like how to improve organizational structure (e.g., Prietula, Carley, & Gasser, 1998), public policy research about how predictions of behavior can influence policy (e.g., Barrett, Eubank, & Marathe, 2006), how to improve building and planning exit approaches in industrial engineering (e.g., Galea, Blake, Gwynne, & Lawrence, 2003; Stewart, Elyan, Isaacs, McEwen, & Wilson, 2017), human-in-the-loop simulations (e.g., Thiruvengada & Rothrock, 2007), and simulating computer generated forces in military simulation (e.g., Bolton, 2013; Morgan, Morgan, & Ritter, 2010; Surdu & Parsons, 2006). Neural network models are also not included because they tend to be used to model shorter time-span behavior than is useful for system design.

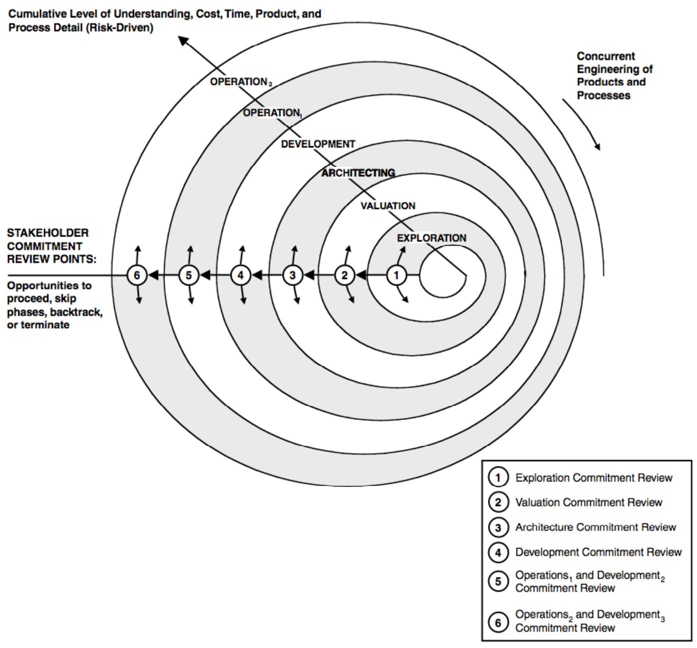
2. Useful features for using models of cognition in system design

So, how can cognitive models be used in system design? There are several ways. A primary way, of course, is to help the designer understand and predict a major system component: the user.

A way to organize how models of human cognition can be used in system design is to take a theory of system design and note where models can be used. The risk-driven spiral system development approach (Pew & Mavor, 2007) is such an theory. This approach emphasizes a spiral approach to design, that of checking with stakeholders every iteration through envisioning. With this in hand, then individual uses and tools for manipulating models are described.

Risk-driven spiral system development approach

The risk-driven spiral system development approach (Pew & Mavor, 2007) suggests developing a user model at each step of the design and concurrent development of the user model as the system is developed. This approach is shown in Figure 1. In this approach, the model is typically seen as a cognitive, rather than a physical, model because the important user aspects are seen as cognitive rather than physical; however, for some system design, a physical or integrated model may be more appropriate.


Cumulative Level of Understanding, Cost, Time, Product, and Process Detail (Risk-Driven)
Concurrent Engineering of Products and Processes
OPERATION$_2$
OPERATION$_1$
DEVELOPMENT
ARCHITECTING
VALUATION
EXPLORATION
STAKEHOLDER COMMITMENT REVIEW POINTS:
Opportunities to proceed, skip phases, backtrack, or terminate
1 Exploration Commitment Review
2 Valuation Commitment Review
3 Architecture Commitment Review
4 Development Commitment Review
5 Operations$_1$ and Development$_2$ Commitment Review
6 Operations$_2$ and Development$_3$ Commitment Review

[figure should be inserted from accompanying PDF]

Figure 1. A representation of the risk-driven spiral system development approach. Reprinted with permission from Pew & Mavor, 2007, Figure 2-5, by the National Academy of Sciences, Courtesy of the National Academies Press, Washington, DC.

In the exploration and valuation stages, the model may not even be formalixed by being written down. As the system is developed, the model of the user is developed alongside it. At the architecting and development phases the model might become written down and used to keep track of the design requirements and be used in design. As the system is developed further, it may be desirable to have a running model that can test the interface by performing tasks with the interface.

The model used for testing has to be more formal and might even interact with the interface (this has rarely been achieved in practice, however). The results of the model's interaction in both qualitative and quantitative terms can be used to further drive design. Section 3 provides a summary of existing models, organized in essentially this order—from informal to formal and computational.

Tools for model use

So, with this approach as a baseline for design, what would a system designer need from a cognitive model? A preliminary list is provided in Table 1. These aspects are next described in detail.

Table 1. Aspects of a modeling tool for use in design.

A model builder

Model (task) libraries

A way to connect models to tasks: simulated eyes and hands

A way to run the model numerous times

Graphical and textual summary of the results

Model builder

The first thing a designer would need to use a model in system design would be a tool to create the model. This tool might be text-based or graphically-based. Simple informal models can use word processors. Most of the task analysis models can use word processors or spreadsheets (e.g., Estes, 2005), although there are some tools to support the generation of these models. For descriptive models this tool might also include what the model knows and what the interests and capabilities of the model are. Complex models of task analysis more often come with more support (e.g., IMPRINT, ProCREW, Cogulator, noted below).

Generative models need further tools because they are essentially low level programming languages about how to use the minds low-level components. Yet these computational information processing models rarely have tools to support creating high-level behavior. There have been attempts to create high-level behavior representation languages for generative models (Paik, Kim, Ritter, & Reitter, 2015; Ritter et al., 2006; Ritter & Norling, 2006; St. Amant, Freed, & Ritter, 2005), but they remain preliminary and relatively little used.

Model (task) libraries

It would be useful to have libraries of models or basic skill sets to combine and build upon, for example, how to do arithmetic, how to use a graphical interface, and how to reason about navigation. While these libraries are available for Java and other languages, they are not generally available for user models, although there was a library of default knowledge for Soar.

It appears that combing tasks, particularly in the production rule format that most architectures use, is more complex than code combination in imperative languages (e.g., Java or Lisp). Combining task knowledge sets may introduce or highlight that there are assumptions that were represented implicitly in the task knowldge that are worthwhile to consider but require modeling expertise to address. This could be because the production rules combine interactively an the procedures run serially in imperative languages. This area will be an area for future work.

Eyes and hands

Simple models are not expected to interact with a system, but the model approach or modeler needs to note that the environment looks like and what an interaction occurs.

More complex computational behavior generating models need a way to receive input from the system and pass actions to the system. To do this, they will need simulated eyes and hands (Ritter, Baxter, Jones, & Young, 2000). This has been done several ways, including by passing information about what would have been seen to a model, but the most satisfying method is probably to allow the model to interact directly with the interfaces being tested (St. Amant, Horton, & Ritter, 2007). Doing this routinely remains an area for research and development (Tehranchi, 2018). In more complex environments interaction could include hearing and movement through the environment as well.

A way to run the model numerous times

When a model has a stochastic component (e.g., built in variability (noise) in its memory retrieval or task performance time), interacts with a system that has varied tasks or other stochastic components (e.g., other agents or a variable system response), or is used to test multiple interface variants, the model will need to be run numerous times to understand its behavior (Ritter, Schoelles, Quigley, & Klein, 2011). It would be useful to have the ability to run the model several times and summarize the output. I only know of one reusable tool in this area (Moore Jr., 2011).

Graphic and textual output displays

With the model output from multiple runs in hand, the designer and other members of the design team will need to understand what the model did and the implications for design. This task may include directly understanding what the model did, and it may also include what-if analyses of how well the model performed on two different interfaces to treatments. So, this will include explainablity of the model and summaries of its behavior. Some of these measures will be straightforward, such as mean time per task and error rate, but there are also measures that are more complex or are derived from other measures, such as learning rate (Ohlsson, 2008) and learning based on task distribution (Ohlsson, 1992).

How models can be used in design

The resulting models can be used several ways in design. Early in the spiral of development, they can be used to discuss the types of users and the types of tasks in the system. At this stage, the designers might wish to make changes in the design based on the assumed capabilities, and the size of the team might be modified based on the number or complexity of the tasks.

Later in the spiral, with a more complete model, designers might make sure that all tasks are supported and that all types of users (and their knowledge, skills, abilities, and personal differences) are supported.

Even later in the spiral, designers might run or apply a model to an interface to see if the model can perform the task, learn to perform the task, or not make errors. The model might be applied by hand like in a cognitive walkthrough (Blackmon, Polson, Kitajima, & Lewis, 2002; Polson, Lewis, Rieman, & Wharton, 1992), or it might be applied like an engineering tool as a simulation (e.g., Byrne & Kirlik, 2005; Gray, John, & Atwood, 1993, and several other papers cited here). The model might be used to judge how many users are required on a multi-person task (Booher & Minninger, 2003; Vicente, 1999), or serve as colleagues within a larger team within a simulation (e.g., Ball et al., 2010).

After the model has been applied yet another opportunity arises. The designer might learn from having applied the model and might make different design decisions in the future. So, while the cost of applying a model might be high, it might lead not only to changes in the current system, but in future systems, and be less than re-designing a system that is too hard to use.

Summary

There are several tasks that designers will ask of models. These needs vary by type of model and also by the use of the model. Informal models used early in the design process ask for less support; generative models interacting with nearly complete systems require the most support and would typically be used later in the design process.

3. Types of cognitive models used in design

There are several types of cognitive models that have been used in design. These are presented here roughly in the order of formality. This is based on the taxonomy in Ritter, Baxter, and Chuchill {, 2014 #1638}The least formal are models that are not articulated by designers but used by them or implied by the artifact. These are called implicit models. There are also informal models, such as would appear on a napkin or be simply be written down. Task analysis is a type of model that focuses on what tasks the user will perform. There are some automatic tools that use a combination of these models that deserve their own category. Finally, computational cognitive models that can perform the task.

Implicit models

Most designed systems have been designed with a user in the mind of the designer. In extreme cases, the user model and the use-cases for a given device will be only in the designer's head (or in previous designers' heads when the designer is copying from a previous design). The user model can be inferred by observers or users. Thus, the models are implicit and not explicit.

For example, chairs include implicit models of how they are used. Every chair designer might not measure the shin length of users and might simply copy a design or reason about how a user sits implicitly. But a handle on the back implies that the designer envisioned a use-case that enabled the chair to be picked up or dragged.

Informal models

Models of users can also be informal, but perhaps represented with written descriptions. These models start to be useful when, for example, the range of users is large (or larger than expected) or when the tasks that users perform are larger or more complex than designers might imagine in a single setting or might be able to keep in mind. The use of personas, prototypical users, might also be considered.

For example, we have found that the range of users of university department websites and the information users are seeking are larger than we initially thought, and larger than the designers were considering. We found about a dozen user types and over 100 types of information that may be sought out by different types of users (Ritter, Freed, & Haskett, 2005). Just writing down this type of model can provide guidance for building and managing web sites because they are often too large to keep in mind.

Task analysis approaches

Task analysis methods attempt to note what tasks users are trying to do. These methods sometimes focus on the user and sometimes can focus on the interface. Both cases will describe the actions taken to complete tasks, but the first case will emphasize cognitive tasks whereas the second case will emphasize interaction with specific interface features like buttons or clicks. These two methods are not mutually exclusive, and parallel or merged analyses are not uncommon.

Perhaps the first task analysis method used in computer design was the Keystroke Level Model of Card, Moran, and Newell (1980; 1983). This approach assigns time costs to various keystroke and mouse moves. The time to use an interface is thus the simple sum of the actions required to perform a task.

Card et al. (1983) also introduced a more complex approach, GOMS. GOMS assumes that there may be multiple strategies and allows for some (though minimal) problem solving. Both of these approaches are for experts doing routine, error-free tasks. GOMS assumes a simple cognitive architecture, the Model Human Processor (MHP). It has also been extended to predict working memory load (Estes, 2005).

There are tools to help apply and compute GOMS analyses. GLEAN is one such tool, which ends up like programming GOMS (Kieras, Wood, Abotel, & Hornof, 1995). CogTool (John, Prevas, Salvucci, & Koedinger, 2004) ends up with a more graphic representation of how the interface looks at each step (using screen shots) to document how the GOMS analysis arises. Cogulator (http://cogulator.io/resources.html, Savage-Knepshield, 2014) is a current version. Antetype (http://www.antetype.com/) appears to use ACT-R and its learning equations to work in a similar way.

These models have been often productively used in system design. For example, in a symposium paper, Chipman and Kieras (2004) note an example GOMS analysis that should have led to a redesign. Not fixing the usability problem found with GOMS led to a procurement program being canceled. Project Ernestine is also often referenced (Gray et al., 1993). In this case, a version of GOMS that allowed multiple-tasking showed that a new design with less keystrokes would be slower than the older system because the keystrokes in the new system were more serial with cognition rather than concurrent. The prediction was upheld with user data but at some cost to the company developing the new system.

There are related models that work at a larger grain size, such as IMPRINT and PROCRU (although PROCRU includes a model of dynamic control), reviewed in Pew (2007). Many similar stories of the use of models to help design and millions of dollars for the US Army are reported by Booher and Minniger (2003) for the IMPRINT system.

Light automatic models

I use light automatic models to indicate tools that will test an interface automatically using a lightweight user model. These tools would apply a set of rules to a website to test such things as links being active, pictures having alt tags, and fonts to be a minimum size. Ivory as part of her thesis (Ivory & Hearst, 2001) created a very useful review of what was available for automatic website evaluation tools at that time. The tools in her review showed that making such tools automatic would help design and be attractive to designers based on their ease of use.

Bobby and later website testers in Ivory's review are examples. These systems test websites and directly make suggestions about how to improve the websites. These models show much promise because they are so easy to use, but the user models being tested are extremely simple.

Computational predictive and generative models

The way to describe the most advanced cognitive models has been often difficult. There are two types of computational models, predictive and generative. The predictive models simply predict the time taken to complete the task. Generative models will generate the behavior to complete the task, a process, which typically predicts the time as well. A predictive model might predict the time to do a task, but the task would not include the information processing implied. For example, a predictive model would predict the time to do a large multiplication problem but not the answer or common errors; the generative model would generate the answer, and the time to compute the answer, and perhaps common errors. Both communities may call the models cognitive models, although only the generative models can live or work with a system to perform a task. Tools in the previous section on task analyses often are grouped with predictive cognitive models, a type of non-generative cognitive model.

These models tend to be built as part of cognitive architectures. A cognitive architecture, broadly speaking, is a fixed set of information processing mechanisms used across all tasks (Newell, 1990). Examples include Soar (Laird, 2012), ACT-R (Anderson, 2007; Ritter, Tehranchi, & Oury, in press), CoJACK (Ritter et al., 2012), and EPIC (Kieras, Wood, & Meyer, 1997).

There are several useful examples of computational generative models being used in design. Pew and Mavor (2007) call for their use throughout the design process. Pew and Mavor (1998) show their use in understanding military tactics. The ACT-R community has used them in the design of airport runway systems (Byrne & Kirlik, 2005) and in other ways (Ritter et al., in press, also see act.psy.cmu.edu).

Apex is a slightly different modeling system with a useful, unique perspective (Freed & Remington, 1998). Its approach is not to model the time course of processing, but to illustrate possible errors that could arise. It attempts to find all the significant error pathways; all are assumed to be catastrophic (i.e., in space travel), where the frequency of error is not needed, just that this path could lead to an error. Thus, design should remove any chance of these pathways.

Summary

There have been several types of models of cognition used in system design. One might like to say that one type is better than the other, or to order them in quality. In truth, they have different features, different benefits, and different costs. Some are light weight and easy to use, but do not provide many details about user behavior. Others provide many more details but require more effort to create and use. As Simon Goss said to me once, these cognitive models are models for a brass and mahogany world. Safety critical systems such as flight, power plants, the military are such worlds; high stakes commercial applications probably are as well. Designers will have to choose an appropriate model type based on the system they are designing, resources available to them, and the risks to success that the system is facing.

4. Conclusion

Models of human cognition appear to be important for many design problems. They provide a description of the user's cognition and closely related perception and motor control. There is a wide range of models of cognition used by designers. Designers should use models of users, perhaps created for and based on previous designs, to reduce system risk. They should pay attention to which models to use at which point in the design process.

There remain some open problems for model use in design, which I note next.

Greater usability of models

In nearly every case the models themselves are not seen as easy to use by designers, and thus many designers think that the use of cognitive models in design is poor value. This appears to have several causes. The first is that users are more complex than other components; they can solve problems, learn, and forget, all of which make predicting their behavior more complex. The knowledge that users need to perform tasks is nearly always more complex than it appears to be if you think about it informally. Thus, designers think that the user model will be small and easy and find out in creating it that the model is larger and more complex. This also implies that the knowledge to use the interface is larger and more complex than the designers might anticipate!

It is also the case that the types of information in the models are probably some of the more complex knowledge that we can manipulate and that the usability of modeling tools have not seen as much attention as more widely used tools such as Java. Thus, the usability of models broadly defined is an important area of future work. This view is supported by Pew and Mavor's (2007) National Research Council report, which calls for work in this area.

General usability problems remain, indeed, all the tasks in Table 1, including how to explain the models to lay people, remain problems. It would be useful to have libraries of knowledge to use. These could reduce the cost of creating complex models and help amortize the costs of building a model. Finally, effectively hooking models up to the world remains a problem (that I take up next).

General connection of models to the world

Cognitive models have not been often connected to the world, and should be more often, and this remains true. Early work with cognitive models was about cognition per se, and not about interaction (Gray, 2008). As cognitive models have evolved, they have not been as connected to the world as we might like.

There are several ways for models to interact with interfaces (Ritter et al., 2000). The simplest way so far is to pass the inputs that an interface would provide a human user as reading from a file. This approach is costly in time and quality. It does not allow the world to change based on the model's response, which can limit when this approach can be used. Those models that do interact have, for the most part, interacted in a way that works *around* vision and motor output rather than fully modeling vision and motor output. A more sophisticated and satisfying approach is to interact with the same interface that the user interacts with using the a bitmap representation of the screen and injecting keyboard events to the operating system.

This book represents a useful progression for cognitive models to be used. The problem tying models to tasks and task simulations may be because connecting cognitive models to worlds requires different programming skills than model building (models and interfaces are implemented in different languages and with different programming paradigms). It also requires knowledge of different areas of psychology (motor control and vision vs. cognition). The use of cognitive models to drive models of human bodies offers a new and welcome use that may help realize the potential of both.

References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.

Ball, J., Myers, C., Heiberg, A., Cooke, N. J., Matessa, M., Freiman, M., et al. (2010). The synthetic teammate project. *Computational and Mathematical Organization Science, 16*, 271-299.

Barrett, C., Eubank, S., & Marathe, M. (2006). Modeling and simulation of large biological, information and socio-technical systems: An interaction based approach. In D. Goldin, S. A. Smoka & P. Wegner (Eds.), *Interactive computation: The new paradigm* (pp. 353–394). Berlin Heidelberg: Springer.

Blackmon, M. H., Polson, P. G., Kitajima, M., & Lewis, C. (2002). Cognitive walkthrough for the Web. In *CHI'02 Conference on Human Factors in Computing Systems*, 463-470. New York, NY: ACM.

Bolton, M. L. (2013). Automatic validation and failure diagnosis of human-device interfaces using task analytic models and model checking. *Computational and Mathematical Organization Theory, 19*, 288–312.

Booher, H. R., & Minninger, J. (2003). Human systems integration in Army systems acquisition. In H. R. Booher (Ed.), *Handbook of human systems integration* (pp. 663-698). Hoboken, NJ: John Wiley.

Byrne, M. D., & Gray, W. D. (2003). Returning Human Factors to an engineering discipline: Expanding the science base through a new generation of quantitative methods. Preface to the Special Section. *Human Factors, 45*(1), 1-4.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*(2), 135-155.

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

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

Chipman, S. F., & Kieras, D. E. (2004). Operator centered design of ship systems. In *Engineering the total ship symposium*. NIST, Gaithersburg, MD. American Society of Naval Engineers. Available online at http://handle.dtic.mil/100.2/ADA422107, checked 4 jan 2019.

Elkind, J. I., Card, S. K., Hochberg, J., & Huey, B. M. (Eds.). (1989). *Human performance models for computer-aided engineering*. Washington, DC: National Academy Press.

Elkind, J. I., Card, S. K., Hochberg, J., & Huey, B. M. (Eds.). (1990). *Human performance models for computer-aided engineering*. San Diego, CA: Academic Press.

Estes, S. (2005). Arriving from Delphi at O'Hare: Predicting cognitive engineering in the O'Hare modernization project and beyond. In *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting—2005*, 108-112.

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

Galea, E. R., Blake, S. J., Gwynne, S., & Lawrence, P. J. (2003). The use of evacuation modelling techniques in the design of very large transport aircraft and blended wing body aircraft. *The Aeronautical Journal, 107*(1070), 207-218.

Gray, W. D. (2008). Cognitive architectures: Choreographing the dance of mental operations with the task environment. *Human Factors, 50*(3), 497-505.

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.

Ivory, M. Y., & Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys, 33*(4), 470–516.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of CHI 2004 (Vienna, Austria, April 2004)*, 455-462. New York, NY: ACM.

Kieras, D. E. (1985). The role of cognitive simulation models in the development of advanced training and testing systems. In N. Frederiksen, R. Glaser, A. Lesgold & M. G. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition* (Vol. 22, pp. 365-394). Hillsdale, NJ: Erlbaum.

Kieras, D. E., Wood, S. D., Abotel, K., & Hornof, A. (1995). 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. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230-275.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Moore Jr., L. R. (2011). Cognitive model exploration and optimization: A new challenge for computational science. *Computational and Mathematical Organization Theory, 17*(296–313).

Morgan, J. H., Morgan, G., & Ritter, F. E. (2010). A preliminary model of participation for small groups. *Computational and Mathematical Organization Science, 16*, 246-270.

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

Ohlsson, S. (1992). Artificial instruction: A method for relating learning theory to instructional design. In M. Jones & P. H. Winne (Eds.), *Adaptive learning environments: Foundations and frontiers* (pp. 55-83). Berlin: Springer-Verlag.

Ohlsson, S. (2008). Computational models of skill acquisition. In R. Sun (Ed.), *Cambridge Handbook of Computational Psychology*. Cambridge, UK: Cambridge University Press.

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler *ACM Transactions on Computer-Human Interaction, 22*(5), Article No.: 25.

Pew, R. W. (2007). Some history of human performance modeling. In W. Gray (Ed.), *Integrated models of cognitive systems* (pp. 29-44). New York, NY: Oxford University Press.

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

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press. http://books.nap.edu/catalog.php?record_id=11893, checked Feb 2019.

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies, 36*, 741-773.

Prietula, M. J., Carley, K. M., & Gasser, L. (1998). *Simulating organizations: Computational models of institutions and groups*. Cambridge, MA: The MIT Press.

Ritter, F. E. (1993). *TBPA: A methodology and software environment for testing process models' sequential predictions with protocols* (Technical Report No. CMU-CS-93-101): School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Tech. Report CMU-CS-93-101.

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), 141-173.

Ritter, F. E., Bittner, J. L., Kase, S. E., Evertsz, R., Pedrotti, M., & Busetta, P. (2012). CoJACK: A high-level cognitive architecture with demonstrations of moderators, variability, and implications for situation awareness. *Biologically Inspired Cognitive Architectures, 1*(1), 2-13.

Ritter, F. E., Freed, A. R., & Haskett, O. L. (2005). User information needs: The case of university department web sites. *ACM interactions, 12*(5), 19-27. acs.ist.psu.edu/acs-lab/reports/ritterFH02.pdf.

Ritter, F. E., Haynes, S. R., Cohen, M. A., Howes, A., John, B., Best, B., et al. (2006). High-level behavior representation languages revisited. In *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, 404-407. Trieste, Italy: Edizioni Goliardiche.

Ritter, F. E., & Norling, E. (2006). Including human variability in a cognitive architecture to improve team simulation. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 417-427). Cambridge, UK: Cambridge University Press.

Ritter, F. E., Schoelles, M. J., Quigley, K. S., & Klein, L. C. (2011). Determining the number of model runs: Treating cognitive models as theories by not sampling their behavior. In L. Rothrock & S. Narayanan (Eds.), *Human-in-the-loop simulations: Methods and practice* (pp. 97-116). London: Springer-Verlag.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human performance in synthetic environments: A supplementary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center (HSIAC).

Ritter, F. E., Tehranchi, F., & Oury, J. D. (in press). ACT-R: A cognitive architecture for modelling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*.

Savage-Knepshield, P. (2014). Cogulator, a tool for measuring cognitive workload: Interview with Steven Estes. *HFES Bulletin, 57*(6), 4 pages.

St. Amant, R., Freed, A. R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research, 6*(1), 71-88.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction, 14*(1), Paper 1. 24 pages.

Stewart, A., Elyan, E., Isaacs, J., McEwen, L., & Wilson, L. (2017). The effect of person order on egress time: A simulation model of evacuation from a Neolithic visitor attraction. *Human Factors, 59*(8), 1222–1232.

Surdu, J. R., & Parsons, D. (2006). Army simulation program balances agile and traditional methods with success. *Crosstalk: The Journal of Defense Software Engineering, 19*(4), 4-8.

Tehranchi, F., & Ritter, F. E.. (2018). Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R. In *Proceedings of the 16th International Conference on Cognitive Modeling (ICCM 2018).* 162-167.

Thiruvengada, H., & Rothrock, L. (2007). Time window-based performance measures: A framework to measure team performance in dynamic environments. *Cognition, Technology & Work, 9*(2), 99-108.

Vicente, K. (1999). *Cognitive work analysis*. Mahwah, NJ: Erlbaum.