Tehranchi, F., & Ritter, F. E. (2017). An eyes and hands model for cognitive architectures to interact with user interfaces. In MAICS, The 28th Modern Artificial Intelligence and Cognitive Science Conference. 15-20. Fort Wayne, IN: Purdue University. http://ceur-ws.org/Vol-1964/

An Eyes and Hands Model for Cognitive Architectures to Interact with User Interfaces

Farnaz Tehranchi
Department of Computer Science and Engineering
Penn State, University Park, PA 16802 USA
farnaz.tehranchi@psu.edu

Frank E. Ritter
College of Information Sciences and Technology
Penn State, University Park, PA 16802 USA
frank.ritter@psu.edu

ABSTRACT

We propose a cognitive model to interact with interfaces. The main objective of cognitive science is understanding the nature of the human mind to develop a model that predicts and explains human behavior. These models are useful to Human-Computer Interaction (HCI) by predicting task performance and times, assisting users, finding error patterns, and by acting as surrogate users. In the future these models will be able to watch users, correct the discrepancy between model and user, better predicting human performance for interactive design, and also useful for AI interface agents. To be fully integrated into HCI design these models need to interact with interfaces. The two main requirements for a cognitive model to interact with the interface are (a) the ability to access the information on the screen, and (b) the ability to pass commands. To hook models to interfaces in the general way we work within a cognitive architecture. Cognitive architectures are computational frameworks to execute cognition theories—they are essentially programming languages designed for modeling. Prominent examples of these architectures are Soar [1] and ACT-R [2]. ACT-R models could access the world interacting directly with the Emacs text editor [3]. We present an initial model of eyes and hands within the ACT-R cognitive architecture that can interact with Emacs.

KEYWORDS

Cognitive Model; Cognitive Architecture; Human Computer Interface, Interaction.

1 INTRODUCTION[1]

HCI has used cognitive models of users successfully in different ways: for examining the efficacy of different designs to predict task performance times, helping to create and choose better designs, and saving overall system cost by providing feedback for designers. In the future these models will be able to watch users, have more realistic input, correct themselves, and predict human performance.

To be useful in HCI research, these models need to be capable of interacting with interfaces. The two main requirements for a cognitive model to interact with a task environment are (a) the ability to pass commands, and (b) the ability to access the information on the screen—the cognitive model encodes screen's objects that has been used as an agent architecture. If cognitive models interact with user interfaces, then the models will be easier to develop and apply. This approach, can be called a Cognitive Model Interface Management System (CMIMS) [4], which is an extension of the concept of a User Interface Management System (UIMS) [5].

Cognitive architectures are infrastructures for cognitive science theory and provide computational frameworks to execute cognitive theories. They are programming languages specifically designed for modeling, such as Soar [1, 6] or ACT-R [2]. A user model is thus a combination of task knowledge and a cognitive architecture with its fixed mechanisms to apply the knowledge to generate behavior.

The aim of this research is to develop a cognitive model and provide ACT-R models access to the world by enabling ACT-R to interact directly with the Emacs text editor. This approach has been called Esegman, for Emacs Segman [7], related to another tool, Segman, for hooking user models and agents to interfaces [8]. ACT-R can communicate with a task environment by instrumenting a graphical library [9], in this case the Emacs text editor. Emacs is an interactive text editor that works with spreadsheets [10] and includes extensions to read email and browse the web.

A brief review on the Esegman approach is given in section II. Section III and IV explain the ACT-R structure and our eyes and hands model. Finally, conclusion remarks and future research are discussed in section VI and VII respectively.

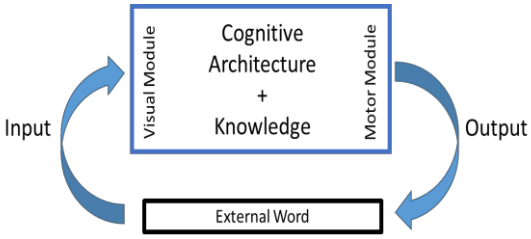

Figure 1: Cognitive model structure.

2 THE ESEGMAN APPROACH

Emacs Segmentation/Manipulation (Esegman) provides a connection between the cognitive architecture and the world. Both ACT-R and Emacs are written in Lisp. It allows us to extend them and design a bridge between them. This bridge will enable ACT-R to communicate with an interactive environment. ACT-R cognitive models have a loop with three components: the task environment, the model, and the results (see Figure 1). The cognitive model receives its perspective as

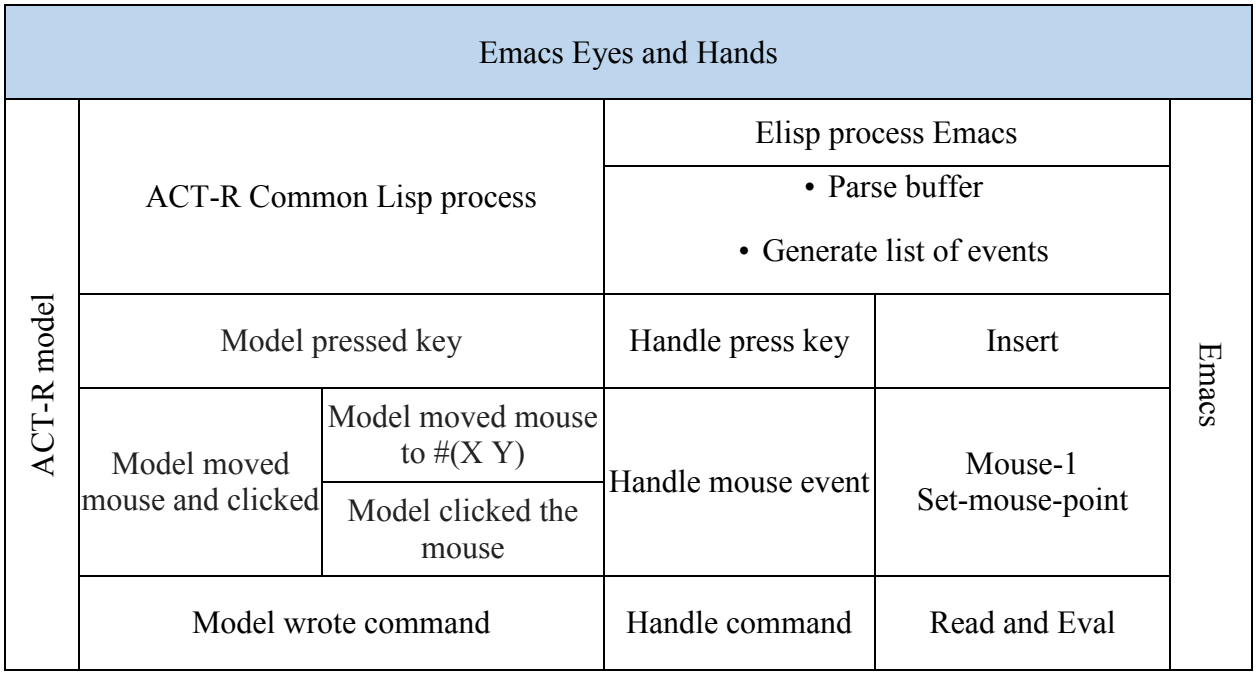

Figure 2: Emacs Eyes and Hands output approach, showing how model process (left) are implemented in the Emacs text editor (right)

input from the visual (and auditory) module and outputs actions through the motor/manual module. The cognitive architecture within this process understands the external world as shown in Figure 1.

The modeling process can start with collecting data from the real world. This data gathering has been performed using the Dismal spreadsheet [10]. Dismal is an augmented spreadsheet developed for Emacs that was designed to gather and analyze behavioral data. Users can interact with the spreadsheet through keystrokes, menus, and function calls. Participants have performed a set of spreadsheet tasks in Dismal [11], and their performance over 14 subtasks, about 20 min. of work, was recorded using the Recording User Input (RUI) software [12, 13]. As will be discussed in the next section, we thus have an existing dataset of human behavior on a complex task to model [11] and also have a model available [14]. A comparison has been done, but not on a fine-grained level, and without modeling interaction, error, and error correction. Because the model did not have access to the spreadsheet, which is complex to model without duplicating a spreadsheet. This model was developed by High-level behaviour representation language (Herbal)—an open source software that supports two cognitive architectures and one agent architecture through a set of common cognitive modeling tasks [15]. Herbal compiles into declarative knowledge and procedural knowledge, and its output has a syntax similar to ACT-R.

In particular, for our connection between the architecture and the world, we will use Emacs functions to take the interaction commands from the model (ACT-R will be a sub-process of the Emacs process) and insert them and their effects into the target Emacs window; and similarly take the state of Emacs and make it available to the model. Figure 2 diagrams most of the components of this approach. Therefore, the model will be able to execute commands and be aware of their results. For example, after collecting the commands from ACT-R and receiving requests to 'look' and move the eye, the courser in Emacs window will move. This approach provides the output direction in Figure 1. However, for input direction of Figure 1 we have to feed ACT-R with the information about the world by inserting contents into its visual module.

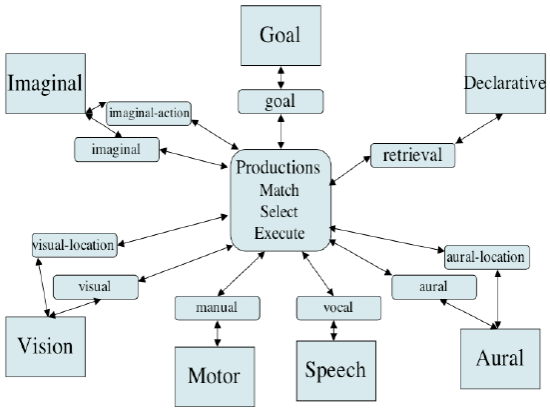

Figure 3: The ACT-R 6 Architecture.

Table 1: Initial Production Rules of the Eyes and Hands Model.

Production Rules & Buffers used	
Start	Goal, Manual, Visual-Location
Rule1	Goal, Visual-Location, Visual
Rule2	Goal, Manual, Vocal
Rule3	Goal, Manual
Rule4	Goal, Vocal

3 THE ACT-R ARCHITECTURE AND MODEL

The two basic types of knowledge in ACT-R are declarative and procedural. Declarative knowledge (Chunks) contains

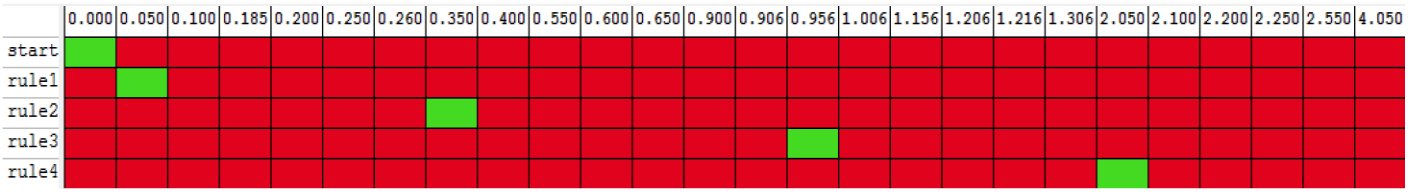

Figure 4: Production History in Different Critical Cycle

facts, images, locations, and sounds. Procedural knowledge (production rules) indicates behavioral aspects of performance with goals, operators, methods, and selection rules [16]. All tasks in the model [14] we will build on include a combination of declarative and procedural knowledge.

Figure 3 shows a schematic of ACT-R's architecture and the default modules of ACT-R [17]. The ACT-R modules communicate through buffers, which may contain a memory Chunk. The default set of modules can be partitioned into Goal, Imaginal, Declarative, Vision, Motor/Manual, Speech/Vocal, and Aural Modules. The model presented in the next section exercises the Manual, Vision, Goal, and Vocal modules. Table 1 shows corresponding buffers.

Buffers in ACT-R are the interfaces between modules. Each buffer is connected to a specific module and has a unique name. A buffer is used to relay requests for actions to its module and a module will respond to a query through its buffer. In response to a request, the module will usually generate one or more events to perform some actions or place a Chunk into the buffer. A module has access to any buffer at any time, but can only manipulate its own buffer [18].

ACT-R runs a pattern matching process that finds production rules with conditions that match the current state of the system—the left hand side of production rules. Then, tasks are performed if conditions match and the actions of production rules will be fired (so-called RHS). Productions can make a request to an ACT-R buffer and then that buffer's module will perform whatever function it provides may then place a Chunk into a buffer (summarized in Table 1 for our model). These production rules can be tested by comparing the performance, time to perform the task, accuracy in the task, of the model performing the task with the results of people doing the same task. Furthermore, we can use neurological data (fMRI) as measures of cognitive psychology.

Herbal/ACT-R generates an agent in ACT-R from a Dismal spreadsheet model represented hierarchically for a sequence of tasks and the relationship among these tasks [14]. The task representation is based upon hierarchical task analysis and GOMS. GOMS (Goals, Operators, Methods, and Selection Rules) is a high-level language in which HCI tasks can be articulated in a hierarchical form to decompose complex tasks into simpler ones [19]. In this GOMS-like cognitive user model the production rules are created from the hierarchical tree structure. The model retrieves the next node/sub-task from the declarative memory elements according to a depth-first search algorithm to complete the task. In this model, the novice model has to carry out more memory retrievals. In contrast, an expert model uses fewer declarative memory elements. We present an initial model of eyes and hands within the ACT-R cognitive architecture that can interact with Emacs in next section that complete the output direction of Figure 1.

```
(p rule1
    =goal>
     isa goal
     step 0
    =visual-location>
    ?visual>
     state free
    ==>
    +visual>
     isa move-attention
     screen-pos =visual-location
    +goal>
     isa goal
     step 1)
```

Figure 5: Sample ACT-R Production.

4 THE EMACS EYES AND HANDS MODEL

In this section we briefly describe a small model used to test and exercise the Esegman architectural additions. In this model declarative memory Chunks are created initially as the basic building blocks. We define two Chunks types, one for the target window location and one for Goal buffer to track the steps of model. This model is a static model without learning and uses the GOMS framework for HCI. The model proceeds as follow. After setting the parameters that control the general operation of the system, we define Chunk types for the model and declaring the configuration of slots that will be utilized in Chunks. The ACT-R has an ability to process visual objects on a computer screen. In this regard, two separate buffers have been defined for locations and objects. The visual object has multiple features such as color, shape, height and width. But visual location is only representing the location of these objects. Thus in ACT-R the location can be retrieved separately [20].

An ACT-R model was constructed in a way that is able to interact with the same experiment software as the human participants. The manual module provides the functionality to interact with Emacs window such as the moving courser, clicking, monitoring hand movement, and use a mouse and a keyboard. However, it requires a hook to facilitate these functionalities.

Figure 4 demonstrates the productions history according to Table 1. The critical cycle in ACT–R is when the buffers hold representations determined by the external world and internal modules, patterns in these buffers are recognized, a production fires, and the buffers are then updated for another cycle [21]. The assumption in ACT–R is that this cycle takes about 50 ms to complete—this estimate of 50 ms as the minimum cycle time for cognition has emerged in a number of cognitive architectures including Soar (Newell, 1990 [1]).

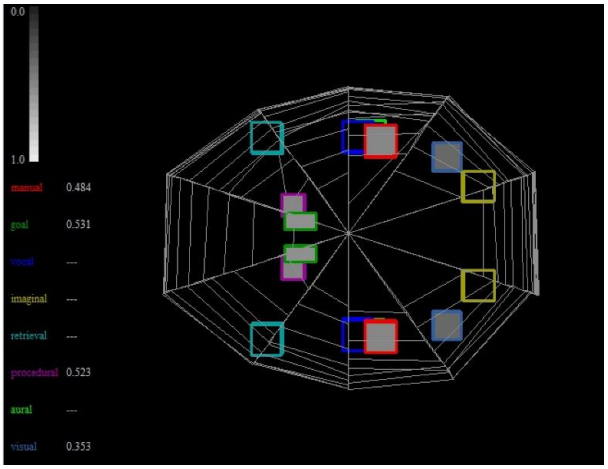

Figure 6: 3D-brain viewer, showing what parts of the brain are active at a point in the model execution.

The first production rule (Start) makes a request to the visual-location buffer. The visual-location module places the location Chunk into a visual-location buffer. Also, in the Start rule the manual module presses the keystroke. The model moves the visual attention and uses the retrieve visual location to reflect the visual object in Rule1. Figure 5 shows a sample production, the equivalent of the Rule1. This production describes operations on three buffers, goal, visual-location, and visual. Each of these buffers has a set of slots, prefixed by the '=', its values can be constrained in a standard way. On the left hand side of the production, the goal buffer is constrained to hold a goal with a given name and slot (here goal 0), and it must be found in memory. Also the visual buffer state should be free and there must be a Chunk in visual-location. On the right hand side, the current task and state slots of the goal buffer are updated (to 1) and the visual module move the attention to the address in visual-location buffer. In Rule2 model moves the mouse courser to the visual object that at the previews task moved the attention to and will click upon it in Rule3. The Ruld4 uses the vocal buffer to speak out the commands. Table 1 shows a summary of all modules and buffers that were called through the corresponding tasks.

Figure 4 shows the different time costs of productions rules. Time columns correspond to when the procedural module attempted to find a production to fire (the matched production). Figure 4 demonstrates the use of serial modules in parallel [22] because of the parallel threads of serial processing in each module. For instance, Rule2 should wait for the completion of Start rule because at 0.35 the action at which Rule2 matches was completed—the manual state will be free again. But, the Rule2 used the three modules Goal, Manual, and Vocal, which can all make a request to their buffers in parallel (three threads). The serial bottleneck of ACT-R is its buffer limitation to a single declarative unit of knowledge, Chunk, thus only a single memory can be retrieved at a time.

The ACT-R model explicitly can be a map to the physical implementation of the mind, the human brain (Figure 6) and how activation of brain areas used by the running model predict activation of the brain in human when solving a problem. Figure 6 was dveloped by ACT-R enviroment and brain associated buffers are displayed. Figure 6 shows a prefrontal region that tracks the operations in the goal buffer, a motor region that tracks operations in the manual buffer, and a vision region that tracks operations in a visual buffer that holds the problem representation.

The model will explain how these components of the mind worked to produce rational cognition. In our model four parts of the brain were active and the total amont of active times is presented in Figure 4.

Table 2: Esegman Capabilities.

Implemented	To be implemented
• Press Key • Move Courser • Click Mouse • Execute Obituary Commands	• Move Fovea (focus of visual attention) • Parse Fovea Area • Track/Find Courser • Find Icon

5 Conclusion

The CMIMSs research area can be exploited to help in the development of cognitive models, agents, and supporting them as users that interact with interfaces. But, it will definitely help elevate testing user interfaces, making this process more approachable and easy enough to do that it can be done and does get done. This approach of using model to test interfaces and even systems as they are built has been call by the national research council [23].

By using the Eyes and Hands model in place of a user, questions about user interface designs such as evaluating designs, changing the interface and examining the effects on task performance can be answered more easily. The Eyes and Hands model as a cognitive model approach can prove the advantages of CMIMS in HCI, and start to realize the use of model in system design [24].

In this work, instead of utilizing any methods to get the most accurate output. We use an approach to get the more human-like output and soon input. We present preliminary implementation of this proposed direction. Table 2 demonstrates the summary of our work. We were able to complete the output direction in Figure 1 and augmented the model to perform all sequence of tasks in Figure 2. We will build upon Paik et al's [14] model of the dismal task. It performs the task and learn, but does not interact with a task simulation.

6 Further Research and Limitations

To have a comprehensive cognitive model we need to add the input direction in Figure 1. This remains to be implemented fully in our model. For the input direction of Figure 1, we will have to feed ACT-R models with the information about the world from Emacs by inserting contents into the architecture visual module. In the future these models will be able to watch users, correct the discrepancy between model and user, and better predict human performance for design. These models will be useful to HCI by predicting task performance and times, assisting users, finding error patterns, and by acting as surrogate

users. Also, useful for building interface agents such as Digital Elfs [25].

The model Esegman can intent with surrogate interfaces through Emacs for a wide range of systems—Emacs includes a text editor but also a web browser, a calculator, a spreadsheet tool, and even an implementation of the natural language system Eliza. Thus, we can explore a wide range of behaviors with this approach.

When we extend Esegman to include motor control mistakes, we will also have to include knowledge to recognize and correct these mistakes. This will requires breaking architecture to create human-like mistakes, knowing how to correct it. The model will also have to make mistakes in correction, and in noticing mistakes. They are relatively new types of behavior and knowledge in models, so we think we will learn a lot.

Therefore, adding error analysis to the design will merge our model with large models. However, ACT-R deliberately included limitations such as only a single memory can be retrieved at a time, only a single object can be encoded from the visual field, or only a single production is selected at each cycle to fire can make this approach challenging.

ACKNOWLEDGMENT

This work was funded partially by ONR (N00014-11-1-0275 & N00014-15-1-2275). David Reitter has provided useful comments on Emacs and Aquamacs (the Emacs version for the Mac). We wish to thank Jong Kim who provided the idea for ESegman and Dan Bothell for his assistance.

REFERENCES

[1] A. Newell, *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press, 1990.

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

[3] F. Tehranchi and F. E. Ritter, "Connecting Cognitive Models to Interact with Human-Computer Interfaces," in *Proceedings of ICCM - 2016-14th International Conference on Cognitive Modeling*, University Park, PA: Penn State, 2016.

[4] F. E. Ritter, G. D. Baxter, G. Jones, and R. M. Young, "User interface evaluation: How cognitive models can help," in *Human-computer interaction in the new millenium*, J. Carroll, Ed., ed Reading, MA: Addison-Wesley, 2001, pp. 125-147.

[5] B. A. Myers, "User interface software tools," *ACM Transactions on Computer-Human Interaction,* vol. 2, pp. 64-103, 1995.

[6] J. E. Laird and A. Nuxoll. (2003, Soar Design Dogma. Available at http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/dogma.pdf

[7] J. W. Kim, F. E. Ritter, and R. J. Koubek, "ESEGMAN: A substrate for ACT-R architecture and an Emacs Lisp application," in *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, Trieste, Italy, 2006, p. 375.

[8] R. St. Amant, T. E. Horton, and F. E. Ritter, "Model-based evaluation of expert cell phone menu interaction," *ACM Transactions on Computer-Human Interaction,* vol. 14, p. 24 pages, 2007.

[9] F. E. Ritter, G. D. Baxter, G. Jones, and R. M. Young, "Supporting cognitive models as users," *ACM Transactions on Computer-Human Interaction,* vol. 7, pp. 141-173, 2000.

[10] F. E. Ritter and A. B. Wood, "Dismal: A spreadsheet for sequential data analysis and HCI experimentation," *Behavior Research Methods,* vol. 37, pp. 71-81, 2005.

[11] J. W. Kim and F. E. Ritter, "Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important," *Human-Computer Interaction,* vol. 30, pp. 1-33, 2015.

[12] U. Kukreja, W. E. Stevenson, and F. E. Ritter, "RUI—Recording User Input from interfaces under Windows and Mac OS X," *Behavior Research Methods,* vol. 38, pp. 656–659, 2006.

[13] J. H. Morgan, C.-Y. Cheng, C. Pike, and F. E. Ritter, "A design, tests, and considerations for improving keystroke and mouse loggers," *Interacting with Computers,* vol. 25, pp. 242-258, 2013.

[14] J. Paik, J. W. Kim, F. E. Ritter, and D. Reitter, "Predicting user performance and learning in human-computer interaction with the Herbal compiler," *ACM Transactions on Computer-Human Interaction,* vol. 22, p. Article No.: 25, 2015.

[15] J. Paik, J. W. Kim, and F. E. Ritter, "A preliminary ACT-R compiler in Herbal," in *Proceedings of ICCM - 2009- Ninth International Conference on Cognitive Modeling*, Manchester, England, 2009, pp. 466-467.

[16] J. W. Kim, R. J. Koubek, and F. E. Ritter, "Investigation of procedural skills degradation from different modalities," in *Proceedings of the 8th International Conference on Cognitive Modeling*, Oxford, UK, 2007, pp. 255-260.

[17] F. E. Ritter, M. J. Schoelles, L. C. Klein, and S. E. Kase, "Modeling the range of performance on the serial subtraction task," in *Proceedings of the 8th International Conference on Cognitive Modeling*, Taylor & Francis/Psychology Press, 2007, pp. 299-304.

[18] D. Bothell. ACT-R 7 Reference Manual. Available at act-r.psy.cmu.edu/wordpress/wp-content/themes/ACT-R/actr7/reference-manual.pdf

[19] B. E. John and D. E. Kieras, "The GOMS family of user interface analysis techniques: Comparison and contrast," *ACM Transactions on Computer-Human Interaction,* vol. 3, pp. 320-351, 1996.

[20] D. Peebles and C. Jones, "A model of object location memory," in *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, Austin, TX, 2014.

[21] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological Review,* vol. 111, pp. 1036-1060, 2004.

[22] M. D. Byrne and J. R. Anderson, "Serial modules in parallel: The psychological refractory period and

perfect time-sharing.," *Psychological Review,* vol. 108, pp. 847-869, 2001.

[23] R. W. Pew, "Some history of human performance modeling," in *Integrated models of cognitive systems*, W. Gray, Ed., ed New York, NY: Oxford University Press, 2007, pp. 29-44.

[24] R. W. Pew and A. S. Mavor, Eds., *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 March 2012, 2007.

[25] M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom*, et al.*, "Intelligent agents for interactive simulation environments," *AI Magazine,* vol. 16, pp. 15-40, 1995.