# Building Large Learning Models with Herbal

**Jaehyon Paik[1], Jong W. Kim[2], Frank E. Ritter[3], Jonathan H. Morgan[3], Steven R. Haynes[3], Mark A. Cohen[4]**

[1]Department of Industrial and Manufacturing Engineering
[3]College of Information Science and Technology
Pennsylvania State University, University Park, PA
[2]Department of Psychology, University of Central Florida, Orlando, FL
[4]Business Administration, Computer Science, and Information Technology, Lock Haven University, Lock Haven, PA

<jaehyon.paik, frank.ritter, jhm5001>@psu.edu, jwkim@mail.ucf.edu, shaynes@ist.psu.edu, mcohen@lhup.edu

**Abstract**

In this paper, we describe a high-level behavior representation language (Herbal) and report new work regarding Herbal's ACT-R compiler. This work suggests that Herbal reduces model development time by a factor of 10 when compared to working directly in Soar, ACT-R, or Jess. We then introduce a large ACT-R model (541 rules) that we generated in approximately 8 hours. We fit the model to learning data. The comparison indicates that humans performing spreadsheet tasks appeared to start with some expertise. The comparison also suggests that ACT-R, when processing tasks consisting of hundreds of unique memory elements over times spans of twenty to forty minutes, may have problems accurately representing the learning rates of humans. In addition, our study indicates that the spacing between learning sessions has significant effects that may impact the modeling of memory decay in ACT-R.

## Introduction

In this paper, we discuss the rapid development of user models capable of dynamically representing behavioral constraints. Pew and Mavor (eds., 2007) advise using such user models as a shared representation meant to identify, predict, and when possible, mitigate risks. These representations are of various kinds (qualitative, quantitative, analytical, computational), and can describe interactions operating within or across multiple levels of analysis. These models in their various forms have proven useful in predicting and preventing significant losses whether human (e.g., Byrne & Kirlik, 2005; Pew & Mavor, 2007) or monetary (e.g., Gray, John, & Atwood, 1993) or both (e.g., Booher & Minniger, 2003).

There is a rich literature in user models. Classic user studies beginning with Card, Moran, and Newell's (1983) book have often represented psychological/behavioral constraints using the GOMS model; analyzing user behavior in terms of goals, operators available for accomplishing those goals, routinized sequences of behavior or methods, and rules for the selection of methods for instances where multiple methods apply. Grey et al. (1993) extended and validated the GOMS model through an empirical study of telephone operators working for the New England Telephone Company, introducing CPM-GOMS. The success of later-implemented versions of the GOMS model (John & Kieras, 1996; Kieras, Wood, Abotel, & Hornof, 1995), TAC-AIR Soar (Jones et al., 1999), and of embodied cognitive architectures generally (Byrne, 2001; Byrne & Gray, 2003; e.g., Ritter & Young, 2001; St. Amant, Horton, & Ritter, 2007) has intensified interest in agent-based user models for testing interfaces and for working in simulations as opponents and colleagues.

On the other hand, these efforts have been stymied in part by the significant integration costs and the detailed level of specification required by existing cognitive architectures to create models. While one of cognitive modeling's great strengths is its demand for computational entailment, the low-level abstractions required by mature cognitive architectures such as Soar and ACT-R have frequently proven expensive to create, resulting in a fewer models being created. Furthermore, these models have often proven difficult to maintain, extend, or merge (Pew & Mavor, 1998; 2007; Ritter et al., 2003).

Recognizing these issues, developers in recent years have released both re-implemented versions of Soar and ACT-R in Java that may be easier to integrate into systems, as well as creating high-level cognitive modeling languages that seek to provide a common framework and formal language for a variety of essentially similar cognitive modeling tasks (a review is available, Ritter et al., 2006). In the next section, we will briefly review these efforts before introducing Herbal (High-Level Behavior Representation Language). We will then discuss recent work on Herbal's ACT-R compiler and a large learning model we have generated and tested before concluding.

## Related Work

Cognitive architectures realized as programming languages, as noted above, have operated at low-levels of abstraction, and consequently have made developing, implementing, and comparing cognitive models difficult. Two general approaches have emerged to address this problem, the reimplementation of existing languages and the development of high-level cognitive languages for these architectures. We will describe both briefly before discussing Herbal.

## Re-implementing cognitive modeling languages

Reimplementation of existing cognitive languages into newer object-oriented languages offers several advantages: (a) smoother integration into systems created in those widely used languages, such as Java, supported by extensive libraries and tools; (b) a perceived and sometimes greater degree of implementation modularity, and thus the ability to more easily investigate changes and extensions to existing cognitive architectures; and (c) the opportunity to make comparative analyses, and thus discern the effect that previous implementation choices as opposed to theoretical commitments have had on the language in question. jACT-R (Harrison, 2002) and jSoar (Ray, 2009) have both contributed interesting comparative analyses, offer an array of GUI based debugging and organizational tools, and can increasingly support ongoing work in simulations and agent-based tools.

jACT-R and jSoar both rely heavily upon the syntax of their parent languages to represent the rules and knowledge, limiting their accessibility to some extent. Though Python ACT-R (Stewart & West, 2005) eliminates this syntax issue, all three languages are at various stages of completeness, and none to our knowledge has undergone extensive validation through a computational alignment or docking study (Axtell et al., 1996) or similar means (e.g., Burton, 1998; Louie, Carley, Haghshenass, Kunz, & Levitt, 2003). In addition, re-implemented cognitive modeling languages are neither able to support the comparative analysis of models across cognitive architectures, nor the fine-tuning of architectures at a constant high-level of abstraction. Thus, high-level cognitive modeling languages are attractive.

## High-level cognitive modeling languages and approaches

High-level cognitive languages use abstractions to generalize common structures and processes found in existing cognitive architectures. These persistent commonalities are evident when one considers defining a high-level knowledge representation, building a structured task analysis, or implementing a decision cycle characterized by the perceive-decide-act mechanism (Newell, Yost, Laird, Rosenbloom, & Altmann, 1991). Cognitive architectures' shared dependence upon *least commitment* (or the making of control decisions at every decision point) and *associative encoding* (or the associative retrieval of potential courses of action and a conflict resolution process for choosing between solution paths) entail a set of core commonalities from which to abstract. The commonalities include: a declarative memory structure and retrieval method, goals, procedural memory frequently used for the achievement of those goals, mechanisms for responding to external events, and a iterative decision process (Jones, Crossman, Lebiere, & Best, 2006).

Where these approaches differ is in their representation structures. We will briefly summarize two existing candidate approaches for modeling more complex cognitive models: Jones et al.'s (2006) High Level Symbolic Representation Language (HLSR), and Herbal, a High-Level Behavior Representation Language (Cohen, Ritter, & Haynes, in press; Haynes, Cohen, & Ritter, 2009).

HLSR uses three primitives (relations, transforms, and activation tables) to derive micro-theories for representing cognitive architectures (and by extension, cognitive theories). Herbal characterizes common cognitive modeling tasks such as task analyses and problem solving using an ontology based upon the Problem Space Computational Model (PSCM, Newell et al., 1991). Each of these approaches is promising; each potentially allows for comparative analysis across architectures; and each, if fully developed, could promote model reuse across a diverse community of users.

Herbal's user focus, however, is unique in this area. HLSR supports both Soar and ACT-R, but is not yet available outside of its developers, and has, to our knowledge, not undergone either a docking or a usability study. Herbal, in contrast, is open source; supports three cognitive architectures across a set of common cognitive modeling tasks (Soar, ACT-R, and Jess); has undergone two usability studies (Cohen 2008; Cohen, Ritter, & Haynes 2009); has been used to create several models; and is currently undergoing a docking study. Next, we will describe Herbal and work related to Herbal more fully, focusing on Herbal's implications for HCI and the more rapid creation of user models.

## Herbal

Herbal is based on the PSCM (Newell et al., 1991). Herbal's ontological representation defines behavior as the movement of operators modifying states, as well as movement through problem spaces. Within this framework, behavior is divided into bands of activity operating across three time scales: the elaboration cycle (10 ms), the decision cycle (100 ms), and activity occurring within a problem space (1 s). The elaboration cycle describes the process by which an agent modifies its state representation through the associative retrieval of information. The decision cycle in turn consists of repeated cycles of elaboration that persist until quiescence, or until no further productions can be fired. The levels of elaborations are, for the most part, hidden in and by Herbal.

The agent makes decisions based upon its state interpretation and preferences, choosing either a unique operator (actions capable of transforming the state) or generating an impasse if an operator cannot be selected due to insufficient knowledge. Agents resolve impasses by generating sub-states that enable the agent to retrieve the information necessary to specify the next operator. Problem spaces are thus representations describing a sequence of decisions (or a search in the event of limited knowledge) that can be further defined in terms of goals, states, and operators.

Herbal's ontology characterizes behavior in terms of classes that represent concepts such as states, operators, elaborations, impasses, conditions, actions, and working memory. These classes furthermore entail basic relationships for instance—states can contain impasses, working memory, operators, elaborations, and other states while operators and elaborations can contain zero or more conditions and actions. Programming in Herbal thus involves instantiating objects using these ontological classes. Herbal also supplies additional attributes that enable future developers to discern the intent motivating creation of a given object, supporting models that in essence explain themselves (Haynes, Cohen, & Ritter, 2009).

## The Herbal/ACT-R compiler

We have created an initial version of an ACT-R compiler in Herbal. Although several easy-to-use frameworks exist to develop ACT-R models: CogTool (John, Prevas, Salvucci, & Koedinger, 2004), ACT-Simple (Salvucci & Lee, 2003), and G2A (St. Amant, Freed, & Ritter, 2005), these tools cannot represent models of greater complexity than KLM-GOMS or GOMS models.

To support modeling in ACT-R, we added a declarative memory component to the Herbal environment because ACT-R uses declarative memories. With this component, we were able to also add hierarchical and sequential tasks to an ACT-R model—the relations among tasks are shown in a tree form in the user interface. Herbal then makes memories and production rules based on these relationships. Furthermore, to explore the flexibility of the high-level compiler, we added an ACT-R parameter pane. Through this pane, users can generate either a novice ACT-R model or eleven kinds of expert ACT-R models with varying degrees of expertise ranging from 0% to 100%.

The Herbal/ACT-R compiler takes the PSCM representation in Herbal and creates an ACT-R model from it. The compiler also uses these parameters to determine how to compile the model: as a novice, an expert, or somewhere in between. When implementing a task, we represented the level of expertise, or degree of proceduralization, as corresponding to the percentage of declarative memory retrievals necessary to complete the task. We then distinguished novice from expert models by this percentage. Novice models, in this framework, have no information regarding the next task step in procedural memory, and thus must retrieve each step from memory, whereas the expert models have the next task step incorporated as part of the operation. Novice models thus provide the maximum anticipated completion time while *normative expert* models (described below) provide the hypothetical minimum time.

Distinguishing novice from expert, we further divided expert models into two types: (a) *normative experts*, models where all the declarative memory elements for the task have been compiled into procedural knowledge, and (b) *practicing experts*, models that exhibit varying degrees of proceduralization. Models exhibiting 100% expertise (*normative experts*) provide a baseline, and do not use memory elements in declarative memory to perform the task because we assume that and the model has these elements fully proceduralized. Models ranging between 0% and 90% expertise (*practicing experts*) have a proceduralized task structure, but the number of declarative memory retrievals to walk the task structure varies. For example, if a model should be represented as having 10 declarative memories (DMs), the 0% expertise model would have 10 DMs while the 10% expertise model would have 9 DMs and 1 rule, and so on. *Practicing expert* models thus provide us a basis for making useful comparisons with the human data by providing incremental predictions of performance based upon expertise, and perhaps enable us to isolate the participants' actual average level of expertise at the onset of the trial.

## A test of the Herbal/ACT-R compiler

To explore the Herbal ACT-R compiler, we implemented an ACT-R model using Herbal and compared the performance times with practice provided by the model with those of human participants performing the same series of tasks.

### The Dismal spreadsheet task

We next provide a brief description of the participant data before discussing the model and its implications. For the purposes of examining variance in retention rates over time, Kim (2008) devised a sequential spreadsheet task consisting of 14 subtasks that participants learned using one of two different modalities (keyboard or vertical mouse). In addition, Kim examined what if any influence training intervals have on retention rates by comparing the performance of participants undergoing training at 6, 12, and 18-day retention intervals. These results are discussed in a forthcoming publication.

For this comparative analysis, we used a subset of Kim's data, modeling the decline in task completion times for all 14 subtasks over a four training sessions. Over the training iteration's time course (about 30-45 min. per session), Kim found that the average task completion time for participants ($N$ = 30) using a vertical mouse to perform the spreadsheet task ranged from 1,366 s ($SE$ = 60.76 s) on day 1 to 655 s ($SE$ = 22.81 s) by day 4.

The change in performance over the four-day trial is as anticipated, a relationship between performance and practice ($y = 1339.7x^{-0.5}$, $R^2$ = 0.99) that follows the power law of learning. Examining the curve's progression, one also sees the final value is similar to the anticipated KLM-GOMS (Card, Moran, & Newell, 1983) value of 797.14 s for expert performance.

### Modeling the Dismal spreadsheet task

Paik and Kim, working collaboratively, implemented the spreadsheet task model in ACT-R in four hours using Herbal. The resulting novice model consists of 9 rules and 542 declarative memory elements; the fully expert model consists of 541 ACT-R rules and no declarative memory elements; and the intermediate, practicing expert models interpolate between these two models. For example, the

50% expert model has 271 declarative memory elements and 541 ACT-R rules, and the 0% expert model has 542 declarative memory elements and 541 ACT-R rules.

The rate of development using Herbal was about 0.9 minutes per rule (240 minutes x 2 programmers/541 rules) in the expert model. This is approximately 20 times faster than writing in Soar (assuming that an ACT-R rule is approximately the size of two Soar rules to propose an operator and then to implement it), and also about 5 times faster than rates reported by Yost (1992) using TAQL (again, assuming that the ACT-R rules are larger). Unfortunately, we know of no comparative usability studies for ACT-R, but we suspect that Herbal also accelerates the rate of developing ACT-R models.

## Results

Running the 12 models in ACT-R 6, we confirmed that the novice, expert, and intermediate models perform the task. We compared the performance rates over time provided by the model with those of human participants performing the same task. (In addition to the ACT-R model's times, we added interaction times for mouse moves and key presses). All the models also learn, with the novice models learning the most and the expert model the least. The predicted times are comparable both to the GOMS and KLM models, and to the data collected by Kim (2008).

Because the data was taken over multiple trials, the comparison becomes more interesting because we can use the model to predict the participants' levels of expertise at the onset of the first trial. By comparing the learning curves of the model with that of the 40 participants who performed the Dismal spreadsheet task (as depicted in Figure 1), we found that human completion times for the first trial corresponded with an expertise level of 20%, 60% at trial 2, 80% at trial 3, and a gradual increase up to full expert by the fourth trial. We thus see that the human performance data represents a faster learning curve than that displayed by any of the ACT-R models.

The difference between the two curves indicates that the model's learning rate remains too slow, as opposed to the participants' expertise being either too high or too low to be matched. Though the learning displayed by the model is already surprisingly fast and robust, these results suggest not only that the model will have to learn faster but also that it may have to include a new learning mechanism. For example, the learning rate exhibited by the human data between the first and second trials shows a sharp decline, meaning that the participants acquired more knowledge in the first trial than the model, and that this learned knowledge was already sufficiently activated to use for performing the task. The current Herbal/ACT-R models, in contrast, predict a more gradual learning curve. While the existing model includes declarative strengthening and procedural learning, another type of learning or stronger parameters on the existing learning mechanisms may be necessary.

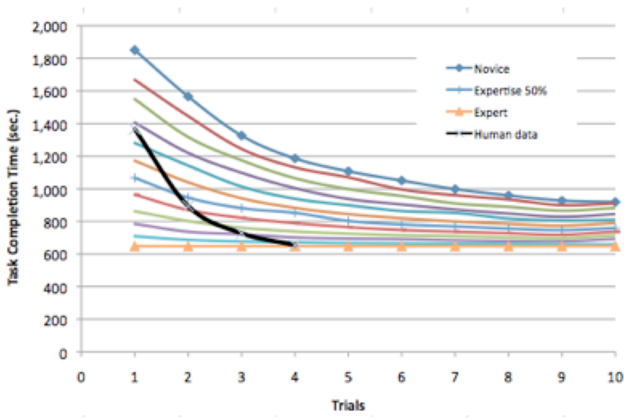

Figure 1: The human data shown with respect to the model's learning curves.

The results in Figure 1 suggest further, deeper problems as well. ACT-R does not appear to easily support modeling declarative memory decay, and the participants' learning sessions were separated by at least a day. If, for example, we attempted to model massed training (concentrated training blocks), the difference between the model's performance and that of the participants is likely to be have been even greater because no memory decay would have occurred, and if we modeled the effect of days between learning trials, the model would learn more slowly, matching the data less well.

Nevertheless, this model is unique in that we are able to begin to conduct these comparative analyses, and perhaps may eventually be better able to ascertain the participants' actual initial level of expertise for sequential tasks. Future work includes individual data fits, exploring these deep problems of decay, and devising ways to achieve faster learning.

## Discussion and conclusions: Implications for future user models

To conclude, we would like to discuss four implications for modeling learning. First, the novice model and the expert models use different approaches to organize the task knowledge that then results in different task completion times. Novice models use a tree structure to organize declarative memories (each declarative knowledge element has a parent, a next-sibling, and a first-child); to walk this structure, the model uses a depth-first search approach. All the expert models, however, use a sequential representation of the declarative memory structure, in other words each of the declarative memory elements has its next step, so the models can walk through the entire structure by following the next step. This is the difference between the top 2 lines, novice and 0% expert models, in Figure 1.

Second, our model's representation of expertise differs from ACT-R. We represented expertise as a function of model's number of declarative memory elements. For example, the 0% expertise model, our novice, has 542 declarative memory elements while our normative expert model (100% expertise) has no declarative memory elements. Consequently (at least as presently compiled), the number of retrieved declarative memory elements gradually decreases as expertise increases. The normative expert model, thus, does not retrieve its declarative memory elements to perform the task. ACT-R, on the other hand, represents experts with production compilation (the process of generating a new rule by combining two or more rules). So, the number of fired rules gradually decreased, but those rules still need to retrieve declarative memory to perform a task.

Third, we have presented a high-level cognitive modeling language that allows for the rapid development of complex user models. As we noted in the introduction, one reason why agent-based user models have not been more widely adopted is because of the relative difficulty associated with developing them. Cognitive architectures such as ACT-R and Soar use a low-level knowledge representation language that makes developing user models appear intractable to non-experts. Herbal, in contrast, is based on the Eclipse that is well-known development tool and provides graphical user interface, so it enables users to make three different kinds of cognitive models, such as Soar, Jess, and ACT-R, more easily. In addition, Herbal provides models that explain themselves by providing answers to questions that users frequently ask (Haynes, Cohen, & Ritter, 2009).

Nevertheless, we acknowledge that Herbal is far from mature, and that we will most likely have to refine our ontology further to fully support ACT-R. We also have to extend the Herbal/Soar compiler to use the task hierarchy pane; and we have yet to compare Soar and Jess models developed in Herbal to human data.

Fourth, the models we have developed with Herbal suggest new model types and new uses for models. A model (Herbal/Soar/Diag) includes a large number of strategies (M. B. Friedrich, 2008; M. B. Friedrich & Ritter, 2009). Another model (Herbal/ACT-R/Dismal) is perhaps the largest ACT-R model (as measured by rule count) created thus far. It is large partially because it performs a non-repetitive task. Many previous models have performed a repetitive task taking minutes to do (e.g., processing 100 planes). Doing a long non-repetitive task, however, requires creating a large knowledge set that has many components that are only used once.

While Herbal remains in some ways a modest step, it opens up new modeling approaches where a broad range of relatively shallow knowledge is needed, but within a cognitive architecture, and where learning is important.

## Acknowledgements

This work was supported by the U.S. Office of Naval Research (ONR) under contract N00014-06-1-0164, N00014-09-1-1124 and the Defense Threat Reduction Agency under contract 1-09-1-0054.

## References

Axtell, R., Axelrod, R., Epstein, J. M., & Cohen, M. D. (1996). Aligning simulation models: A case study and results. *Computational and Mathematical Organization Theory, 1*(2), 123-141.

Burton, R. (1998). Validating and docking: An overview, summary and challenge. In M. Prietula, K. Carley & L. Gasser (Eds.), *Dynamics of organizations* (pp. 215-228). Menlo Park, CA: AAAI.

Byrne, M. D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41-84.

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.

Cohen, M. A., Ritter, F. E., & Haynes, S. R. (in press). Applying software engineering to agent development. *AI Magazine*.

Cohen, M. A., Ritter F. E., & Haynes S. R. (2009), Evaluating design: A formative evaluation of agent development environments used for teaching rule-based programming. In proceedings of the Information Systems Education Conference 2009, 1542-7382, Washington , DC

Cohen, M. A. 2008. A Theory-Based Environment for Creating Reusable Cognitive Models. Ph.D. diss., College of Information Sciences and Technology, The Pennsylvania State Univ., University Park, PA.

Friedrich, M., Cohen, M. A., & Ritter, F. E. (2007). *A gentle introduction to XML within Herbal*. University Park, PA: ACS Lab, The Pennsylvania State University.

Friedrich, M. B. (2008). *Implementierung von schematischen Denkstrategien in einer höheren Programmiersprache: Erweitern und Testen der vorhandenen Resultate durch Erfassen von zusätzlichen Daten und das Erstellen von weiteren Strategien (Implementing diagrammatic reasoning strategies in a high level language: Extending and testing the existing model results by gathering additional data and creating additional strategies).* Faculty of Information Systems and Applied Computer Science, University of Bamberg, Germany.

Friedrich, M. B., & Ritter, F. E. (2009). Reimplementing a diagrammatic reasoning model in Herbal. In *Proceedings of ICCM - 2009- Ninth International Conference on Cognitive Modeling*. Manchester, England.

Harrison, A. (2002). jACT-R: Beta and beyond. In *The 9th Annual ACT-R Workshop*. Pittsburgh, PA.

Haynes, S. R., Cohen, M. A., & Ritter, F. E. (2009). Designs for explaining intelligent agents. *International Journal of Human-Computer Studies, 67*(1), 99-110.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction, 3*(4), 320-351.

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. ACM: New York, NY.

Jones, R. M., Crossman, J. A. L., Lebiere, C., & Best, B. J. (2006). An abstract language for cognitive modeling. In *Proceedings of the 7th International Conference on Cognitive Modeling*, 160-165. Erlbaum: Mahwah, NJ.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

Kieras, D. E., 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. ACM: New York, NY.

Kim, J. (2008). *Procedural skills: From learning to forgetting*. Department of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, PA.

Louie, M., Carley, K. M., Haghshenass, L., Kunz, J. C., & Levitt, R. E. (2003). Model comparisons: Docking OrgAhead and SimVision. In *NAACSOS Conference 2003*, Day 3, Electronic Publication, Pittsburgh, PA.

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

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

Ray, D. (2009). jSoar: A pure Java implementation of Soar. In *The 29th Soar Workshop*. Ann Arbor, MI.

Ritter, F. E., Haynes, S. R., Cohen, M., 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. Edizioni Goliardiche: Trieste, Italy.

Ritter, F. E., & Young, R. M. (2001). Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Studies, 55*(1), 1-14.

Rosenbloom, P. S. (2009). Towards a new cognitive hourglass: Uniform implementation of cognitive architecture via factor graphs. In *Proceedings of ICCM - 2009- Ninth International Conference on Cognitive Modeling*, 114-119. Manchester, England.

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Human Factors in Computing Systems: CHI 2003 Conference Proceedings*, 265-272. ACM: New York, NY.

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), 24 pages.

Stewart, T. C., & West, R. L. (2005). Python ACT-R: A new implementation and a new syntax. In *12th Annual ACT-R Workshop*.