An Integrated Cognitive Architecture for Cognitive Engineering Applications

Shi Cao and Yili Liu
University of Michigan
Ann Arbor, Michigan 48109 USA

The increasing complexity of computational cognitive architectures may increase both their modeling capabilities and their difficulty to learn and use as cognitive engineering tools. This paper reports our work dedicated to enhance the usability and the cognitive engineering applicability of a complex computational cognitive architecture called QN-ACTR, which integrates two complementary architectures Queueing Network and Adaptive Control of Thought-Rational. The aim is to provide an easy-to-use interface and intuitive modeling that support both inexperienced and experienced users in using this complex and powerful architecture. The process of model development is greatly simplified with improved visualization and validation methods. The results were examined using heuristic evaluation. The benefits and practice implications are discussed.

INTRODUCTION

Cognitive models can be used to support cognitive engineering. Compared with other forms of cognitive models such as verbal frameworks and pure mathematical models, cognitive architectures are particularly useful for complex cognitive engineering applications, because they unify a wide range of cognitive theories (Newell, 1990) and can computationally simulate human-machine interactions (Byrne & Pew, 2009; Schunn & Gray, 2002). For example, Adaptive Control of Thought-Rational (ACT-R, Anderson et al., 2004), a cognitive architecture that has incorporated many of the theoretical advances of cognitive science over the past decades, has been applied to cognitive engineering analyses of human-machine interactions including airport runway navigation, driving performance, and human-computer interactions (for a review, see Gray, 2008).

In the recent years, cognitive architectures are becoming increasingly integrated and complex in terms of having more components and interactions between components, requiring the use of knowledge description languages, and involving a large number of parameters. This complexity may increase both modeling capabilities and the difficulty to learn and use them as cognitive engineering tools. For example, building useful models in ACT-R requires a considerable amount of training and practice. The basic theory and syntaxes of ACT-R can be learned by reading a seven-unit tutorial and practicing with examples, which are often covered in a seven-day short course. The model description of displays and controls is written in the Lisp language, and therefore a modeler must also gain reasonable Lisp programming skills. Adjusting model parameters could also be very difficult, because the effect of changing a parameter may be buried deep in the text-based output traces. Currently, most users of cognitive architectures are expert researchers of cognitive modeling. The usage among cognitive engineers in the industry is very limited.

To emphasize the need for the usability development of cognitive architectures for cognitive engineering, Pew (2008) pointed out three challenges for researchers in this field, including the needs for (1) simplified model development, (2) better capabilities for articulating and visualizing how the models work, and (3) model validation.

Recently, several efforts have been made to address these challenges. G2A (Amant, Freed, & Ritter, 2005) and ACT-Simple (Salvucci & Lee, 2003) were developed to automatically translate GOMS (Goals, Operators, Methods, and Selection rules) style operators into ACT-R production rules. Incorporating ACT-Simple, CogTool (John, Prevas, Salvucci, & Koedinger, 2004) simplified the construction of human-computer interaction tasks, allowing the modeling of web browsing tasks by user demonstration with the mouse and the keyboard. Integrating ACT-Simple and an ACT-R driving model (Salvucci, 2006), Distract-R (Salvucci, 2009) simplified the construction of models for human interaction with in-vehicle devices in driving scenarios. Using Visual Basic Application in Excel, a click-and-select user interface has been developed in Queueing Network-Model Human Processor (QN-MHP, Wu & Liu, 2008). It allows users to build QN-MHP models without learning any simulation language. Usability tests showed that this click-and-select interface can save time and reduce errors in model development (Wu & Liu, 2009). In addition, easy-to-use user interfaces have also been developed in E-GOMS (Gil, 2010) and SANLab-CM (Patton & Gray, 2010).

The previous efforts have simplified model development, reducing or eliminating the need for learning a modeling language. However, each of them still has limitation in one or more of the following aspects.

- The simplification of modeling work comes at the cost of limiting the task displays and controls that can be model to a limited set of tasks.
- The flexibility to construct customized models with customized parameters is limited, for example, not being able to define the road curvature of each road segment as in a human driving experiment.
- Human information processing that can be modeled is limited to procedural and perceptual-motor processes, lacking the capabilities to model complex cognitive tasks such as learning, decision making, and sentence comprehension.

 DOI 10.1177/1071181312561075

- The visualization of how the model works can be improved to include both the visualization of mental information processing and task interaction.
- Model simulation and its validation with human experiments use different platforms. The potential discrepancies between human and model tests (display and control mechanisms) may confound the validation results.

Considering these aspects where improvements can be made, the work reported in this paper developed the usability of an integrated general-purpose cognitive architecture, QN-ACTR (Figure 1), which integrates two complementary cognitive architectures Queueing Network (QN, Liu, Feyen, & Tsimhoni, 2006) and ACT-R. It combines QN's advantages in modeling multitask performance and mental workload and ACT-R's advantage in modeling complex cognition. It has been demonstrated that the integrated architecture can generate the same modeling results given the same model setup as in ACT-R (Cao & Liu, 2011a) and can model mental workload using server utilization (Cao & Liu, 2011b).

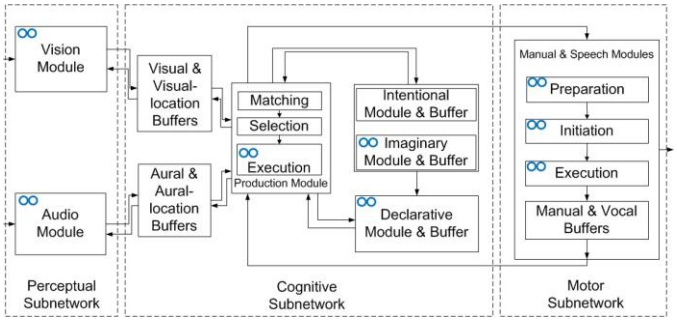

Figure 1. Server structure of QN-ACTR. Queue symbols (two circles) mark the servers with queues.

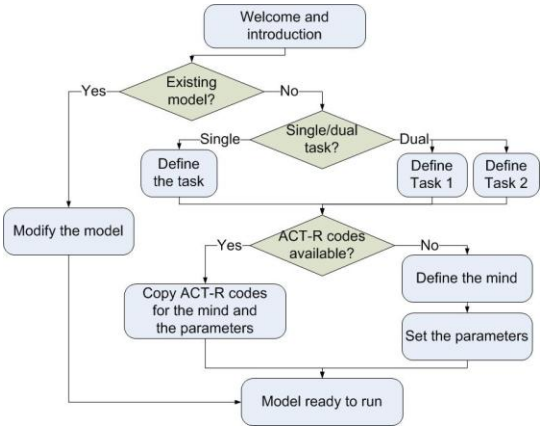

Figure 2. Flow chart of model development in QN-ACTR.

In QN-ACTR, the model setup includes three parts: the task, the mind, and the parameters (Figure 2). The mind part includes both the procedural knowledge of production rules and the declarative knowledge of chunks. The mind and the parameter parts use identical syntaxes as in ACT-R.

This paper reports the completed usability development in QN-ACTR that addresses Pew's three challenges (2008), including (1) the two methods of building a model to simplify model development, (2) the visualization of modeling results to articulate how the model works, and (3) an integrated human experiment interface for model validation. These features are evaluated using Nielsen's ten heuristics for user interface design (1994), which is one of the best-known methods of usability assessment (Hollingsed & Novick, 2007). The benefits and practice implications are discussed.

PRACTICE INNOVATION

Further developed on the basis of a previous version (Cao & Liu, 2011a), QN-ACTR is implemented in Micro Saint® Sharp, a C#-based discrete event simulation software package, which is also the platform for IMPRINT (Allender, Kelley, Archer, & Adkins, 1997). Micro Saint® Sharp provides built-in features that support the usability development of QN-ACTR such as easy-to-use model setup and visualization.

Simplified model development

Two methods for model setup have been developed in QN-ACTR: a basic method using text-based syntaxes and a click-and-select interface.

The basic syntax method supports fast and direct model editing (i.e., copy and paste), which is designed for advanced users. Among the three parts of a QN-ACTR model (i.e., the task, the mind, and the parameters), syntaxes for the mind and the parameters are the same as in ACT-R. ACT-R syntaxes for the task are essentially Lisp codes, which supports the modeling of a wide range of tasks but requires users to learn the Lisp programming language. To simplify task building while keeping the capability to model a wide range of tasks, we have developed QN-ACTR task syntaxes that include different templates to model typical tasks. A task template is a general description for a type of experiments. A modeler can build a task by setting the template's parameters according to the experiment setup. For example, the Day-Block-Trial template can be used to define discrete-event experiments. This template has been used to replicate 17 models from the ACT-R 6.0 (v1.3) tutorial and three dual-task models from threaded cognition (Cao & Liu, 2011a). The World3D template can be used to define dynamic tasks such as driving. When combined, the two templates can define a dual-task of driving and another mental task such as arithmetic computation. These modeling syntaxes cover both static tasks, where display stimuli are not affected by previous responses, and dynamic tasks, where display stimuli are affected by previous responses.

Although the syntax method has its advantages, it may take a long time for novice users to learn. To support novice users, we have developed a click-and-select interface named Model Setup Assistant (MSA) that can help user generate model syntaxes. MSA is programmed in C# and added to Micro Saint® Sharp as a plug-in module. Providing both the syntax and the click-and-select interfaces can meet the needs of both novice and advanced users. This dual-mode method has been implemented in scientific software such as SPSS®.

Similar to the user interface in QN-MHP (Wu & Liu, 2009), MSA in QN-ACTR allows users to define a model by

writing texts in tables and selecting options from menus. Users start from selecting the single or dual task scenario or loading demonstrations (Figure 3). Previous research models are included as demos (samples), such as ACT-R Tutorial models, driving models, and a transcription typing and reading dual-task model. New models can be made by modifying existing demos.

When a single task is selected with a template such as the Day-Block-Trial template for discrete-event experiments or the World3D template for driving, a task setup window will appear, asking users for the information needed in the experiment setup. When a dual-task is selected, two windows will appear each defining a task. For example, the Day-Block-Trial template in MSA asks for configuration settings such as whether a display stage in a trial will be terminated when all responses are detected (Figure 4) and setup details such as the number of trials in a block and the type of a display item (e.g., text, line, button, and tone) (Figure 5). The World3D template defining a driving task asks for road and car details such as lane width, road type, road length, and other cars' speed (Figure 6).

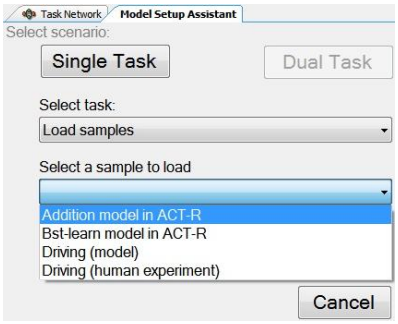

Figure 3. Screenshot of selecting a task using Model Setup Assistant.

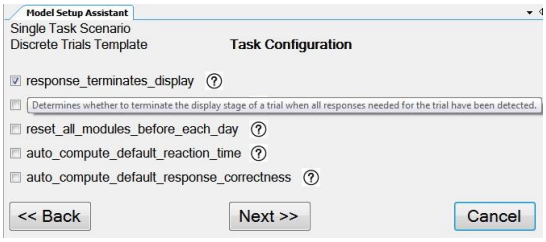

Figure 4. Screenshot of selecting task configuration options using Model Setup Assistant.

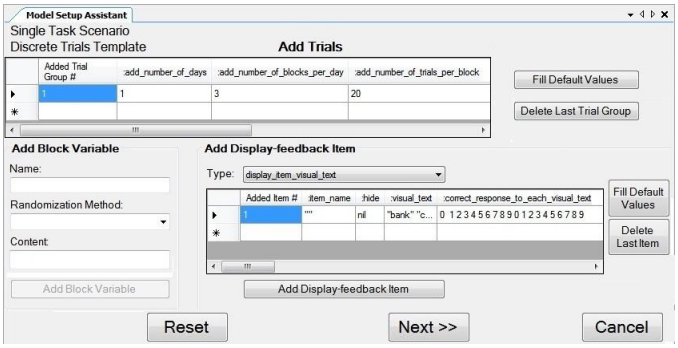

Figure 5. Screenshot of defining a discrete task using Model Setup Assistant.

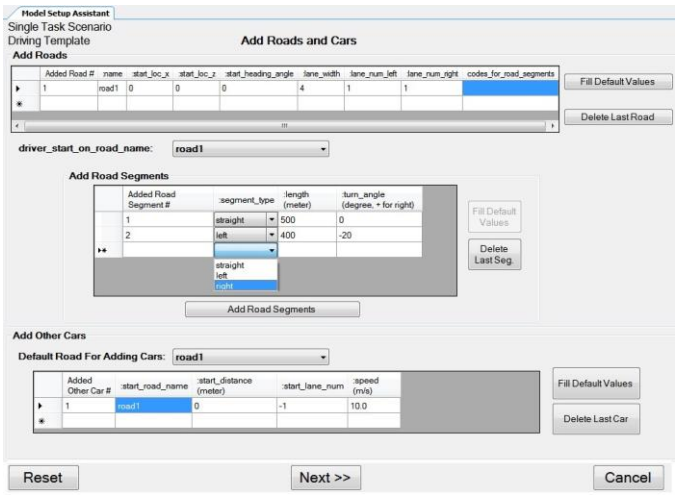

Figure 6. Screenshot of defining a driving task using Model Setup Assistant.

After defining the task, MSA can also assist users to define the mind and the parameter parts of a model. Since these syntaxes in QN-ACTR are the same as in ACT-R, if existing ACT-R codes are available, users can simply copy the ACT-R codes and paste into a QN-ACTR model. If no existing code available, users can define the mind, including chunks (Figure 7a) and production rules (Figure 7b), and set parameters (Figure 7c) with the assistance of MSA, by filling in tables and selecting from lists without the need to learn the knowledge description language used in ACT-R.

The model generated by MSA is also written in syntaxes. The resulted syntaxes can be saved, edited, or directly used to run the model. Simple modification of a model such as changing a few parameters can be easily achieved by directly editing the syntax file.

Visualization of 3D dynamic tasks

Previous work has developed the visualization of mental information processing, discrete experiment displays and controls, and the multi-dimensional mental workload (Cao & Liu, 2011a, 2011b). A new feature added to the visualization capabilities of QN-ACTR in this study is visualizing 3D dynamic tasks.

Using Animator3D in Micro Saint® Sharp, 3D dynamic tasks such as driving in single or dual task scenarios can be visualized in real time while the model is performing the task, which allows intuitive observation of model performance. The system refresh rate can be set by the user (10 ms by default). System dynamics such as speed, steering angle, and lateral deviation are visualized and recorded. Figure 8 illustrates that the model is driving a car while performing an arithmetic addition task. The model is following the car in the right lane and is visually focusing on the car. At the same time, the model is speaking "three" in response to the question of "1 + 2," which is displayed through the auditory channel and visualized on the right hand side of the figure.

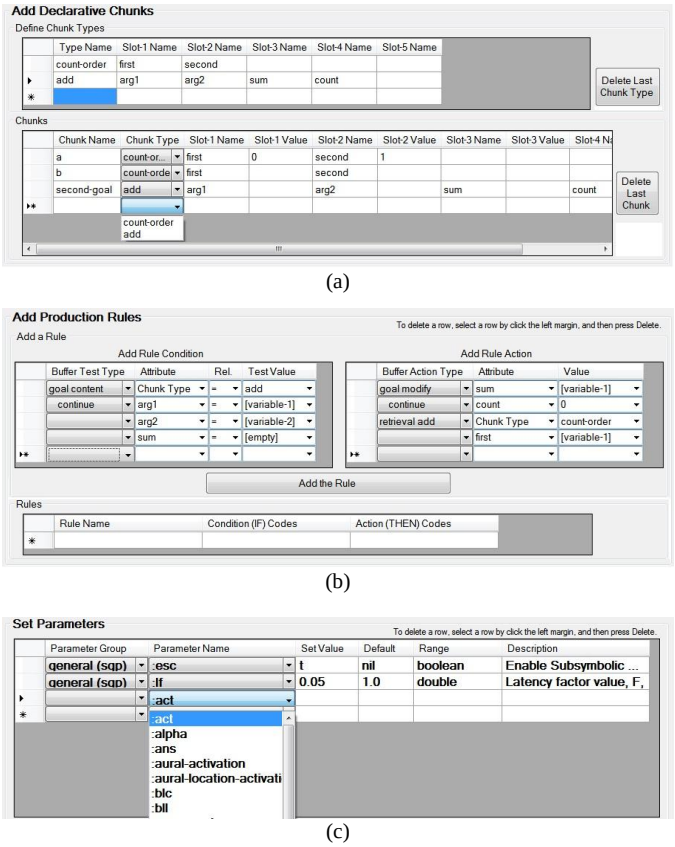

Figure 7. Screenshot of defining the mind and the parameter parts of a model using Model Setup Assistant, including (a) chunks, (b) production rules, and (c) parameters.

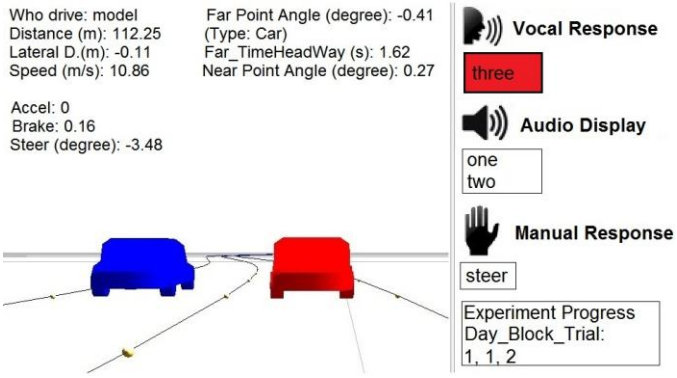

Figure 8. Visualization of a driving and arithmetic addition dual-task in QN-ACTR.

Integrated human experiment interface

The same task interface with which the model interacts can also serve as the interface for human participants to complete the same tasks. We have developed a human driving interface in QN-ACTR that supports simulated driving experiments with steering wheels and pedals. This feature allows the model and the human to perform and be compared in the same tasks with identical interfaces, with no need to replicate the real world experiment system in the modeling platform for the model to interact with (e.g., Salvucci, 2006). Using the same experiment platform avoids any discrepancy between human and model tests due to the experiment setup.

FINDINGS

The usability development of QN-ACTR is evaluated using Nielsen's ten heuristics for user interface design (1994).

Visibility of system status. MSA always shows the stage of model development at the top-left corner. The visualization of the mind and the task keeps users informed about what is going on in the model during the simulation. Buttons in MSA are dimmed and disabled when their actions cannot be performed in some cases. Program responses and feedbacks are immediate with no delay.

Match between system and the real world. All the column headers in MSA tables and the items in menus use self-explanatory phrases without abbreviation. The steps of modeling in MSA follow the logical order shown in Figure 2. Full names and detailed descriptions are shown for each abbreviated ACT-R parameter name (Figure 7c).

User control and freedom. MSA supports undo (e.g., change the road name, delete a chunk, and reset a table) and redo (e.g., go back to the previous stage, and then go next again). A cancel button is provided at each stage to exit the setup at any time, and then users can restart MSA if needed.

Consistency and standards. Definitions and names are used consistently throughout all modeling steps. Tables and menus follow similar layouts and styles. Button position is the same between templates and stages.

Error prevention. The use of menu selection in MSA tables prevents the input of invalid items. Table cells automatically perform validation check, and users are notified when an input is of an invalid type or out of the valid range. Duplicated names assigned by users (e.g., chunk names) are automatically revised to prevent run-time errors. Syntax errors are also reported before the simulation starts.

Recognition rather than recall. MSA provides menus for users to select their options and tables to fill in. Model developing knowledge is provided to users in the interface. For example, users do not have to learn any modeling syntax. Instead, they can describe the model in natural language and fill in blanks or select items (Figure 7a, b). The default value, valid range, and description of model parameters are displayed for the users (Figure 7c).

Flexibility and efficiency of use. The syntax method and MSA cater to both inexperienced and experienced users. Experienced user can speed up the modeling work by directly copying and editing syntaxes. Syntaxes for the mind and the parameters can also be directly copied from ACT-R codes.

Aesthetic and minimalist design. MSA tables and menus are organized and aligned in groups. Introductions and explanations are concise.

Help users recognize, diagnose, and recover from errors. Error messages are expressed in plain language (no codes) and precisely indicate the problem. For example, "Error! Set General Parameters needs para_name: :lf to be a double rather than: nil."

Help and documentation. Help information is embedded in MSA. For example, the corresponding help information is shown when the mouse rests on the question mark beside a task configuration item (Figure 4). A QN-ACTR user manual has also been developed to provide detailed instructions.

DISCUSSION

A cognitive architecture is both an integrated theory of cognition and a computerized simulation platform that can be used for cognitive engineering applications. The complexity of cognitive architectures allows the modeling of complex cognitive mechanisms, but at the same time, increases the difficulty to learn and use them as cognitive engineering tools. The work reported in this paper developed the usability of an integrated general-purpose cognitive architecture, QN-ACTR, which integrates two complementary cognitive architectures QN and ACT-R. The aim is to provide easy-to-use modeling for both inexperienced and experienced users while keeping the capability to model a wide range of tasks.

This paper reports the usability development in QN-ACTR that addresses the three challenges in human performance modeling (Pew, 2008). The completed work includes (1) the two methods of building a model to simplify model development, (2) the visualization of modeling results to articulate how the model works, and (3) an integrated human experiment interface for model validation.

Evaluated using heuristic evaluation for user interface design (Nielsen, 1994), the usability development of QN-ACTR has met the usability guidelines. In particular, Model Setup Assistant, a click-and-select user interface, simplifies model development and allows model setup by selecting from menu items and filling in blanks. Users can describe the model following natural language and experiment logic without the need to learn any programming or cognitive engineering language. Auto-check functions have also been developed to prevent modeling errors. In addition, a new feature of visualizing 3D dynamic tasks is added to the visualization capabilities of QN-ACTR. Dynamic tasks such as driving in single or dual task scenarios can be visualized in real time, which allows intuitive observation of model performance.

A human experiment interface has also been integrated into QN-ACTR, which allows the model and the human to perform and be compared in the same tasks with identical display and control setups. This method helps eliminate the need to replicate the real world experiment system in the modeling platform for the model to interact with and therefore avoids any discrepancy between human and model tests due to the experiment setup.

In conclusion, the usability development of QN-ACTR supports easy-to-use modeling for cognitive engineering applications. It allows users who are not experts of cognitive modeling to explore its application in human factors tests and evaluation. Since QN-ACTR uses the same syntaxes as ACT-R to define chunks and production rules and set parameters, Model Setup Assistant in QN-ACTR can also generate modeling codes for ACT-R models and help simplify the model development in ACT-R. Future studies will further improve the usability of QN-ACTR as a cognitive engineering tool and examine the results with empirical usability tests.

REFERENCES

Allender, L., Kelley, T., Archer, S., & Adkins, R. (1997). IMPRINT: The transition and further development of a soldier-system analysis tool. *MANPRINT Quarterly, 5*(1), 1-7.

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

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Byrne, M. D., & Pew, R. W. (2009). A history and primer of human performance modeling. *Reviews of Human Factors and Ergonomics, 5*, 225-263.

Cao, S., & Liu, Y. (2011a). Integrating Queueing Network and ACT-R cognitive architectures. *Human Factors and Ergonomics Society Annual Meeting Proceedings, 55*, 836-840.

Cao, S., & Liu, Y. (2011b). Mental workload modeling in an integrated cognitive architecture. *Human Factors and Ergonomics Society Annual Meeting Proceedings, 55*, 2083-2087.

Gil, G.-H. (2010). *An accessible cognitive modeling tool for evaluation of human-automation interaction in the systems design process.* Doctoral Dissertation. North Carolina State University.

Gray, W. D. (2008). Cognitive modeling for cognitive engineering. In R. Sun (Ed.), *The Cambridge Handbook of Computational Psychology* (pp. 565-588). New York: Cambridge University Press.

Hollingsed, T., & Novick, D. G. (2007). Usability inspection methods after 15 years of research and practice. *SIGDOC '07: Proceedings of the 25th Annual ACM International Conference on Design of Communication*, 249-255.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). *Predictive human performance modeling made easy.* Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems, Vienna, Austria.

Liu, Y., Feyen, R., & Tsimhoni, O. (2006). Queueing Network-Model Human Processor (QN-MHP): A computational architecture for multitask performance in human-machine systems. *ACM Transactions on Computer-Human Interaction, 13*(1), 37–70.

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

Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen & R. L. Mack (Eds.), *Usability Inspection Methods*. New York, NY: John Wiley & Sons.

Patton, E. W., & Gray, W. D. (2010). SANLab-CM: A tool for incorporating stochastic operations into activity network modeling. *Behavior Research Methods, 42*(3), 877-883.

Pew, R. W. (2008). Foreword. In D. C. Foyle & B. L. Hooey (Eds.), *Human Performance Modeling in Aviation*. Boca Raton, FL: CRC Press/Taylor & Francis Group.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48*(2), 362–380.

Salvucci, D. D. (2009). Rapid prototyping and evaluation of in-vehicle interfaces. *ACM Transactions on Computer-Human Interaction, 16*(2), 9: 1-33.

Salvucci, D. D., & Lee, F. J. (2003). *Simple cognitive modeling in a complex cognitive architecture.* Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems, Ft. Lauderdale, Florida, USA.

Schunn, C. D., & Gray, W. D. (2002). Introduction to the special issue on computational cognitive modeling. *Cognitive Systems Research, 3*, 1–3.

Wu, C., & Liu, Y. (2008). Queuing network modeling of the psychological refractory period (PRP). *Psychological Review, 115*(4), 913–954.

Wu, C., & Liu, Y. (2009). Development and evaluation of an ergonomic software package for predicting multiple-task human performance and mental workload in human–machine interface design and evaluation. *Computers and Industrial Engineering, 56*, 323–333.