Integrating models and tools in the context of driving and in-vehicle devices

Bonnie E. John (bej@cs.cmu.edu)
Human-Computer Interaction Institute, Carnegie Mellon University, 5000 Forbes Ave.
Pittsburgh, PA 15213 USA

Dario D. Salvucci (salvucci@cs.drexel.edu)
Department of Computer Science, Drexel University, 3141 Chestnut St.
Philadelphia, PA 19104 USA

Peter Centgraf (centgraf@cmu.edu), Konstantine Prevas (gusp@cmu.edu)
Human-Computer Interaction Institute, Carnegie Mellon University, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Abstract

The integration of cognitive models offers interesting theoretical and practical benefits to the development of complex models, but at the same time brings up new challenges that the modeling community is only beginning to address. In this paper we present our experiences integrating models and tools for modeling,. Specifically, on the modeling end, we integrated an ACT-R model of driving with an independently-built model of cell phone dialing. To do this, we integrated the driving tool (a minimal-graphics driving simulator that can be controlled by ACT-R) with a modeling-by-demonstration tool, CogTool, that also used ACT-R as its cognitive engine. We found that both the models and tools required re-working because of the integration. The models required new motor movements and methods for switching between tasks. The tools required new specification techniques for placing prototyped devices in the vehicle and demonstration techniques for indicating points of task-switching. Our experiences indicate that while the integration of cognitive models is indeed an approach worthy of pursuit, the field may not understand all the ramifications of such integration until many more integrated models and tools are developed.

Introduction

People often integrate separate skills in the course of their everyday lives. For example, a person learns how to drive separately from learning to use a cell phone, and integrates these two skills when they use a cell phone while driving. Likewise, to make building computational cognitive models of complex domains tractable, a reasonable approach is to integrate independently-built models of different behavior. (Here, we are defining "integration"` of models to mean the joining together of independently-built models of tasks at approximately the same task size, as opposed to the "composition" of models from smaller tasks that are components of a larger task, e.g., the composition of templates at the level of mouse movements or keystrokes into large tasks (John & Gray, 1992; John et al., 2002).)

The integration approach has advantages over constructing complex task models from scratch or even from smaller components. For example, an expert in a particular domain can invest time and expertise in producing a veridical model of an important behavior, and it can be integrated with models of different behavior by other people who are not expert in that domain. The first author took this approach in the early 1990s when modeling the tasks of the NASA Test Director (the person responsible for assuring that all tests are conducted on the Space Shuttle before launch). Because the NASA Test Director's job involves substantial verbal communication, we integrated an independently-developed, in-depth and well-verified model of natural language comprehension (NL-Soar, Lewis, 1999) with a hierarchical goal decomposition model of the NTD's task knowledge (John, Remington & Steier, 1991; Nelson, Lehman, & John, 1994a,b). Even when working within a single cognitive architecture like Soar or ACT-R, the integration approach is not without difficulties (discussed in Nelson, Lehman, John, 1994a), because independently-built models often do not exchange information or pass control in a way that interleaves the tasks appropriately.

Because this approach is relatively new in computational cognitive modeling, each new attempt teaches lessons in how we might proceed with models, methods, and tools to make integration more successful and easier. This paper describes our most recent work in integrating stand-alone Keystroke-Level Models (KLM, Card, Moran & Newell, 1980) of skilled users of information devices with a model of driving an automobile (Salvucci, Boer & Liu, 2001). In this work we integrated both models and modeling tools, to the betterment of both the models and the tools. We will briefly describe the separate components that contributed to this work, then describe our experiences integrating the tools, the resulting models, and an iteration on both the models and the tools.

Separate Contributing Components

The work reported integrates two models, an ACT-R model of driving an automobile (Salvucci, et.al., 2001) and a KLM of dialing a cell phone, and two tools, a "minimally-graphic" simulated driving environment[1] (Salvucci et al., 2001) and CogTool (John et al., 2004). CogTool is a tool for

[1] Salvucci maintains two versions of his driving model for operating a simulator with textured road graphics (see animation at http://hcil.cs.drexel.edu/projects/drivermodel.html) or the "minimally-graphic" simulator used here and shown in Figure 2.

cognitive modeling by demonstration focused on HCI tasks, which itself is an integration of several tools: Dreamweaver™, the Behavior Recorder (Koedinger, Aleven & Heffernan, 2003), ACT-Simple (Salvucci & Lee, 2003), and the ACT-R environment (Bothell, 2004).

ACT-RPM and Salvucci's Model of Driving

The model of driver behavior is an ACT-R model that integrates control, monitoring, and decision making to navigate highway environments with traffic. It uses the full ACT-RPM cognitive architecture, with simulated eyes, hands, and feet as well as ACT-R's cognitive processor. For control, the model employs a two-level model of steering that uses a "far point" on the road to guide predictive steering and a "near point" on the road to center the vehicle. For monitoring, the model encodes its surrounding environment using ACT-R's simulated eyes to maintain situation awareness. For decision-making, the model checks the current situation and decides when to perform maneuvers such as lane changes. Thus, the driver model incorporates both lower-level perception and action for vehicle guidance and higher-level cognition for awareness and decision-making. This driver model has been shown to account for a number of aspects of human highway driving, including nearing the inner curb during curve negotiation and switching gaze to the destination lane at the start of a lane change (see Salvucci et al., 2001).

When the driver model is run in ACT-R, a window opens with a simulated driving environment. This window contains a picture of the simulated environment (i.e., the roadway and other vehicles) from the point of view of the driver (model). The far point and near point are indicated and the model's point of attention and eye fixation are displayed. As the model drives, the road appears to move past the car and the eye fixation point moves as it assesses the driving situation.

CogTool

The CogTool is a new tool for creating KLMs by demonstration (John et al., 2004). The KLM technique uses a very simple framework for modeling skilled performance on a computer-based task. It was originally described in Card, Moran, and Newell (1980) and later in Card, Moran, and Newell (1983, Ch. 8). The KLM makes several simplifying assumptions that provide more constraint than other modeling frameworks in cognitive science. For example, the analyst must specify the method used to accomplish the particular task of interest, which typically entails modeling specific task instances. Furthermore, the specified method is limited to containing only a small set of pre-established keystroke-level primitive operators. Given the task and the method, the KLM uses duration estimates of these keystroke-level operators to predict the time a skilled user will need to execute the task. The original KLM included six types of operators: K to press a key or button, P to point with a mouse to a target on a display, H to home hands on the keyboard or other device, D to draw a line segment on a grid, and R to represent the system response time during which the user has to wait for the system. Each of these operators has an estimate of execution time, either a single value, a parameterized estimate (e.g., K is dependent on typing speed and whether a key or mouse button click, press, or release is involved), or a simple approximating function (e.g., Fitts's Law estimates for P). A final operator, M, represents the pauses sometimes observed between users' actions, presumably to mentally prepare to do that action or a closely-related series of primitive actions. Using a single mental operator instead of decomposing it into different types for different perceptual and cognitive activities was a deliberate simplification. Card, Moran and Newell (1983) explored many different levels of granularity and simplification and determined that this single M was sufficient for many HCI design problems. The KLM also includes a set of five heuristic rules for placing mental operators to account for mental preparation time. An analyst constructing a KLM by hand starts with Rule 0 which places Ms before many Ks or Ps, then applies Rule 1 through Rule 4 to remove many of the Ms just placed. If applied rigorously and consistently, these rules produce a model that matches skilled execution time to well within 20%.

Despite its simplicity, some user interface (UI) designers in Human-Computer Interaction see KLM as a relatively difficult technique to learn and use. Novices creating KLMs by hand often forget to include all physical operators and have difficulty placing Ms rigorously and consistently (John, 1994). CogTool was created in response to these problems, to make it possible for UI designers to create KLMs through demonstration on HTML storyboards. CogTool includes Macromedia Dreamweaver™ extensions that provide a palette of widgets commonly used in interactive devices that can be dragged-and-dropped onto a canvas for WYSIWYG construction of HTML storyboards. In addition, an image can be used as the basis for a storyboard and hotspots inserted in that image to simulate buttons and other interactive devices (see Figure 1). Existing HTML storyboards that were not constructed with our widget palette can be converted to work with modeling-by-demonstration through an "instrument all widgets" command.

After creating the HTML storyboard, an analyst opens the storyboard in Netscape, and demonstrates a task while CogTool's Behavior Recorder (Koedinger et al., 2003) watches the demonstration. An export command then creates a KLM of that task expressed in ACT-Simple, a simplified language that compiles into ACT-R productions (Salvucci & Lee, 2003). The export command inserts Ms (mental operators) automatically, using new rules based on specific widgets to implement Card, Moran and Newell's heuristics (John et al. 2004). The analyst can then load this KLM into ACT-R and run the model. ACT-R operates the HTML storyboard through the Behavior Recorder and produces a time-stamped trace of its activities that predict skilled execution of the task.

Although this particular combination of systems into a single tool for modeling by demonstration is new, it borrows heavily from previous work (CRITIQUE by Hudson et al., 1999 and WorldBuilder reported in Remington et al., 2002). It has been shown to produce valid KLMs for the skilled use

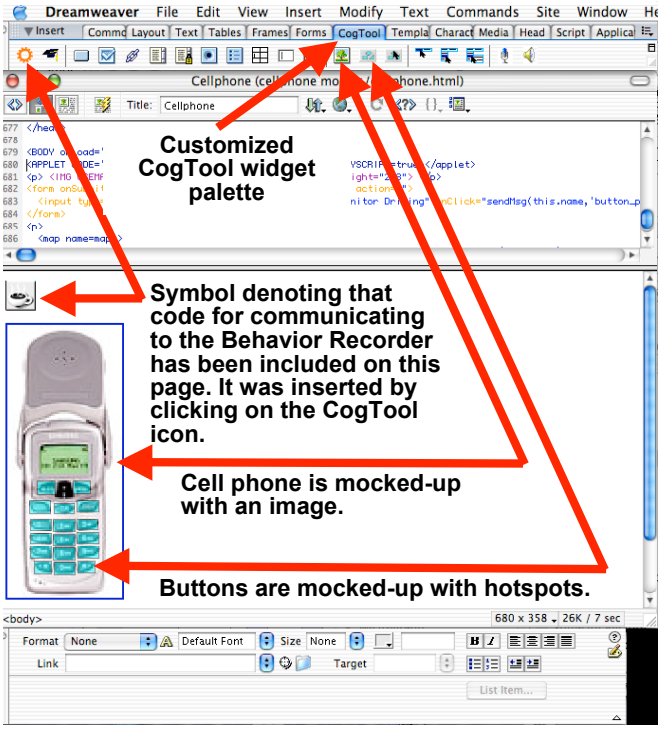

Figure 1: Using CogTool's Dreamweaver™ extensions to mock up a cell phone.

of interactive devices when the CogTool widget palette is used (John et al., 2004).

Storyboards of physical devices typically use an image of the device and hotspots to indicate arbitrary interactive mechanisms, e.g., buttons, knobs and toggle switches. Tasks on such storyboards are more difficult for the CogTool to model because images and hotspots representing different types of mechanisms are all simply tagged as HTML images and hotspots, not as the devices they represent. The CogTool cannot distinguish between them, nor does it possess knowledge about how KLM's rules for inserting mental operators apply to these arbitrary devices. In the absence of such knowledge, CogTool places a mental operator in front of each click on a hotspot, which is likely to over-predict skilled performance times compared to human data for arbitrary devices. To address this problem, we have expanded the CogTool widget set to include some common UI devices that are not standard HTML widgets, e.g., pull-down, cascading menus. Work is continuing to expand valid modeling by demonstration to other common interactive mechanisms found both on computer-based devices and physical devices.

We used CogTool to create an HTML storyboard of the cellular phone used to explore driver distraction in Salvucci, 2001. Demonstrating dialing a telephone number on this image-and-hotspot storyboard produced a stand-alone model of dialing with too many mental operators to match human data, as expected. However, the CogTool produces an intermediate product, namely ACT-Simple code in a text file, which can easily be edited by hand to rectify this problem. The resulting independently-developed model of cell phone dialing is similar to Salvucci's (2001) hand-crafted ACT-R production system.

Integrating the Tools and Models

To make mock-ups of in-vehicle devices as easy to build as mock-ups of desktop interfaces, we enhanced the CogTool so that the models it produces can be integrated seamlessly with the model of driving. Since the CogTool already uses the ACT-R environment, only a few modifications were needed to integrate it with the simulated driving environment.

First, since in-vehicle devices typically use the driver's finger, but the HTML mock-up is demonstrated with a computer mouse, we created an option to export the model as a non-computer-based device. When exporting the model as neither a mouse-based device nor a keyboard-based device, every point-and-click in the demonstration is translated into a simple point based on Fitts's Law, without the click. The original Fitts's Law was derived with measurements where a person tapped a target with a stylus (Fitts, 1954), very similar to a finger tapping the short-travel buttons or touch-screen commonly used on in-vehicle devices. This option is also applicable to modeling interaction with a PDA like a PalmPilot™.

Second, moving to in-vehicle devices required new motor commands in both ACT-R and ACT-Simple. Previously, both languages had operators that homed the hands between keyboard and mouse (KLM's H), pointed with a mouse (KLM's P), and typed with the fingers (KLM's K). To drive while operating an in-vehicle device, we had to add homing between the steering wheel and the device and pressing buttons with the fingers as described above. Though arguably not theoretically interesting, these additions illustrate that new integration and new domains push the development of modeling architectures to cover new classes of human behavior.

Third, an HTML mock-up of a device can be drawn to any scale, but the actual device needs to be placed in the context of the car's dashboard so that eye- and hand-movements can be modeled. Therefore, we needed to record the actual dimensions and location of the in-vehicle device in the storyboard to transmit to the simulator. We added a standard "first-page" to the HTML storyboard with labeled text fields into which the analyst enters the size and location of the device. CogTool reads this first page when the demonstration is exported, creating code in the beginning of the ACT-Simple file that tells the ACT-R environment to place a wireframe of the device in the driving simulation window when the integrated model is run. Figure 2 shows the wireframe of the cellphone's numeric keypad and function buttons on the dashboard console in the lower right.

Resulting Models and Iteration of the Models

Using the integrated tool resulted in an integrated model that ran in ACT-R, displayed the screen shown in Figure 2 so the integrated behavior could be viewed, and predicted relevant measures such as average lateral deviation from the center of the lane and the total time to perform the secondary task

on the in-vehicle device. Unfortunately, when examining the traces and the data produced, the first set of models was insufficient to match human data, or even to be plausible. However, several reasons for this failure immediately presented themselves, which were easy to fix and feed back into the design of the tools.

First, the straightforward integration of the models inherited a decision-making portion of the driving model that switched control from driving to the secondary task (cell phone dialing), when the vehicle's position on the road was reasonably close to the center of the lane and stable. This ability to shift tasks had been included in the driving model when Salvucci hand-crafted the task knowledge to dial a cell phone (Salvucci, 2001). This mechanism worked "out of the box" with the CogTool model. However, the independently-developed model of cell phone dialing did not have any mechanism for switching back to driving. The result was a model that switched to dialing when the driving

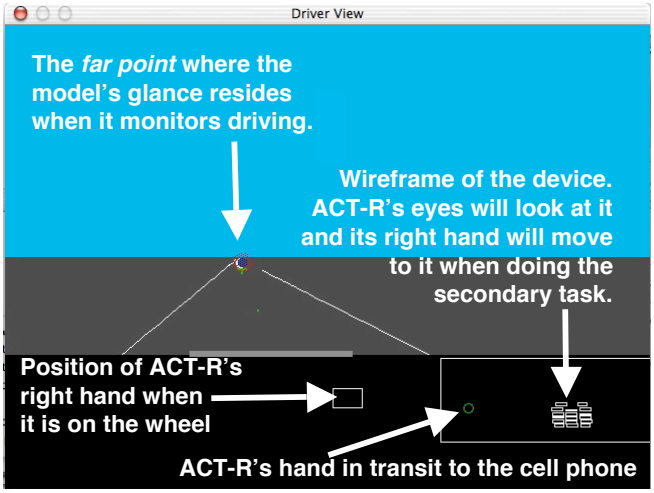

Figure 2. View of ACT-R driving while operating a cell phone mocked up in CogTool.

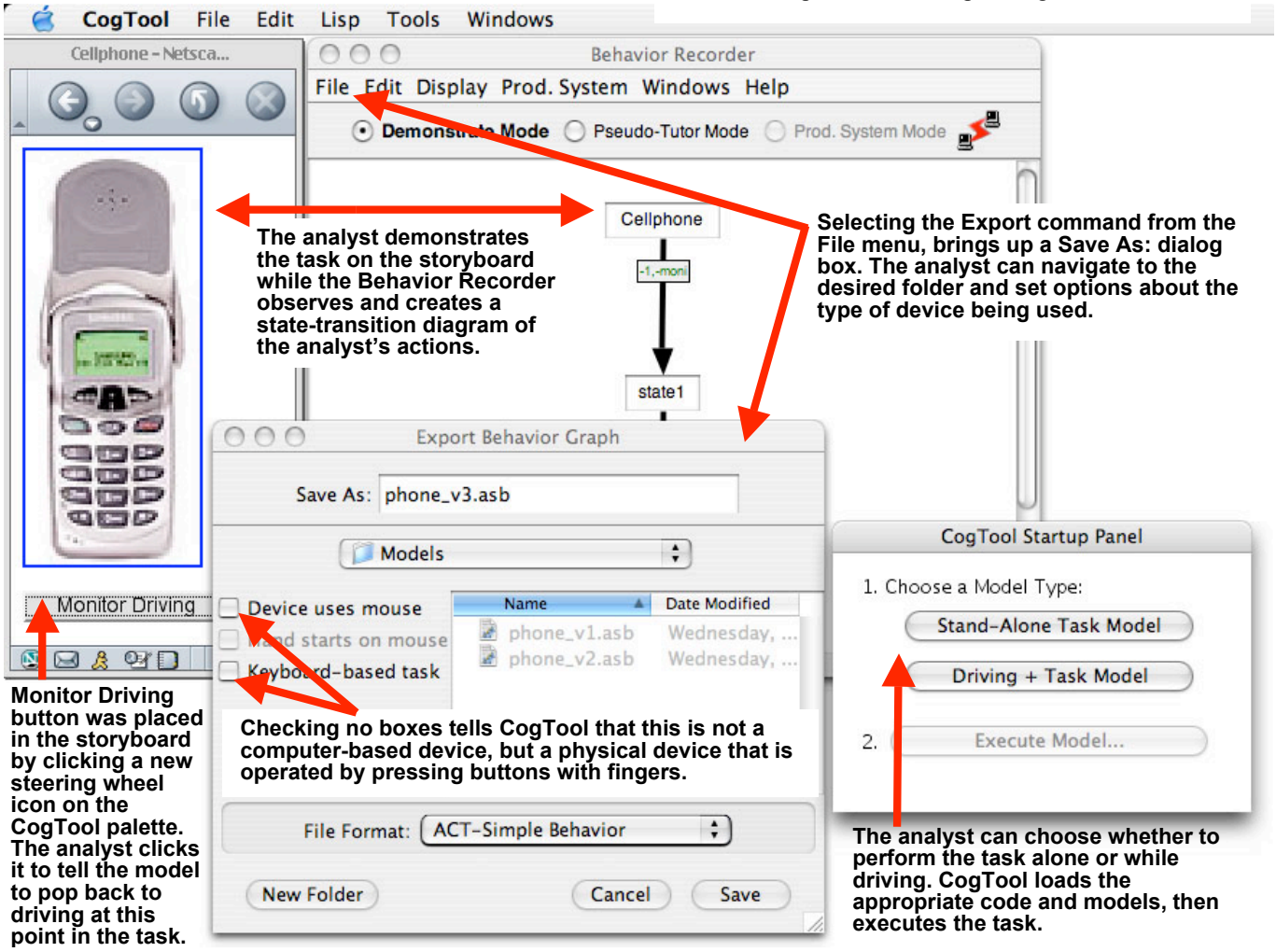

Figure 3. Current version of CogTool being used to demonstrate dialing a phone number on a cell phone. The leftmost window shows the HTML mock-up running in Netscape. The partially-hidden center window shows the Behavior Recorder running and the state transitions it observed in the demo. The window labeled "Export Behavior Graph" in the center is a save dialog box that appears when the behavior observed in a demo is exported to ACT-Simple code (.asb=ACT-Simple Behavior). The rightmost window appears when the Model Launcher portion of CogTool is invoked to let the analyst easily run the model as either a stand-alone task or an in-vehicle information task while driving.

was stable, worked exclusively on that task, and drove off the road! The knowledge of how to interleave secondary tasks with driving in the general case (e.g., Kurosu, 1994) is still under-examined in the field of driver distraction and the subject of further research. However, in response to this problem, we added a "popping" mechanism in ACT-Simple that dictates when the secondary model "pops" back to driving; specifically, the "pop" switches ACT-R's current goal to the driving goal, which runs at least one iteration of vehicle control (i.e., one update of steering angle and pedal depression, taking roughly 200 ms) and then returns switches the goal back to the secondary task. The "pop" can be considered part of a customized executive control process similar to that used by EPIC to switch between tasks in a dual-task environment (see Kieras et al., 2000), or it could be control knowledge that results from learning in a dual-task domain. Both approaches result in behavior similar to our "pop" and match human data in the Wickens' task, a simple dual task (Lallement & John, 1998).

The initial model's lack of knowledge of when to return to driving was compounded by the relatively large mental operators placed by the CogTool. Card, Moran and Newell (1980) used a single type of mental operator, M, with a duration of 1350 ms to denote all operations a skilled user performed that resulted in pauses between keystrokes or mouse movements. Thus, M included visual search, recall, decision-making, verification, and other unspecified perceptual and mental operations. This simplification provided a good fit to keystroke data and was sufficient for an engineering tool to differentiate between different human-computer interaction designs. The models produced by the CogTool implement this M by occupying ACT-R's cognitive processor for 1200 ms and usually launching an eye movement that takes 150 ms (totaling Card, Moran and Newell's 1350 ms). This is again an approximation that is sufficient for evaluating different interfaces when they are simulated as the only task a user is doing. However, when CogTool models are integrated with the driving model, this M takes ACT-R's cognitive processor away from driving far too long for it to maintain a good match to human data. We explored breaking this large mental operator up into smaller pieces that each pop back to driving, working under the notion that people can interleave their mental operations with driving, rather than treating the mental operator as one atomic unit. In this exploration we found that splitting M into three segments of 400 ms each produced a better fit to the human driver data (Salvucci et al., 2004).

Iteration of the Tools

After manipulating the models directly in both ACT-Simple and ACT-R to get a good fit to human data, we can now feed this information back into the design of the tools. For example, the CogTool Dreamweaver™ tool palette now includes an icon that looks like a steering wheel. When this icon is clicked, it places an HTML button labeled "Monitor Driving" at the bottom of the page, outside the boundaries of the mock-up (Figure 3). When demonstrating a task on the mock-up, the analyst can click on the Monitor Driving button to indicate that the device model should relinquish control to driving at this point in the task. This gives the analyst an easy way to control the interleaving of driving and the secondary task.

We have also already included the shorter duration mental operators in CogTool. Now when an M is automatically inserted in a model, it is also automatically broken into three 400 ms segments with a pop back to driving between the segments, as was found to fit best to empirical data (Salvucci et al., 2004).

Because the data on how humans choose to break up secondary tasks and return to driving is not yet conclusive, we wish to make it easy for an analyst to explore many possibilities of when a model pops back to driving from the secondary task. The Monitor Driving button and the sequence of three 400 ms mental operators are two mechanisms already implemented in CogTool for making this easy. However, we intend to make it possible to automatically explore many possible combinations of popping and mental operation durations. Automatic placement of these operations combined with facilities for running and analyzing multiple instances of a task should help researchers quickly explore the space of modeling styles and parameters.

Finally, when more models and data have yielded sufficient information with sufficient confidence, we intend to build heuristics into the CogTool that automatically create models of in-vehicle devices that interleave with driving in a plausible manner.

Conclusions

We conclude from this exercise that integrating independently-built computational models is indeed an approach worthy of pursuit. However, researchers taking this approach in the near future will most likely encounter unforeseen difficulties. Until many more independently-built models are integrated, the field will not understand all the ramifications of integration, nor will it be able to build integrated tools that support this style of modeling in general.

In both the NASA Test Director model mentioned in the introduction and the in-vehicle models described here, the issue of how control is passed from one model to the other arose in unforeseen ways. This issue is likely to continue to be a major challenge until either a theory of general task integration is developed, validated, and incorporated into integration tools, or we convince ourselves that such a general theory is not viable and many examples of domain-specific solutions populate our field. As a third possibility, models may be developed of how people learn to integrate two well-practiced tasks, as was investigated by Chong (1997). In any case, the integration approach to cognitive modeling clearly has both scientific and engineering implications, and will continue to spur improvement and development of theoretical cognitive architectures and also application to real-world complex tasks.

Acknowledgments

This research was supported by grants from the Office of Naval Research (N00014-03-1-0086 & N00014-03-1-0036) to the first and second authors (respectively), a research

contract between General Motors and the first author, and a gift from Ford Motor Company to the second author. The views and conclusions contained in this document are those of the authors and do not represent the official policies, either expressed or implied, of General Motors, Ford Motor Company, the Office of Naval Research or the U. S. Government.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Erlbaum

Bothell, D. (2004) *ACT-R Environment Manual*. Available for download at http://act-r.psy.cmu.edu/software/

Byrne, M. D. & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.) *The Atomic Components of Thought,* (pp. 167-200). Mahwah, NJ: Lawrence Erlbaum.

Card, S., Moran, T., & Newell, A. (1980). The Keystroke-Level Model for user performance time with interactive systems. *Communications of the ACM, 23*, 393-410.

Card, S. K., Moran, T.P. and Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA (1983).

Chong, R. & Laird, J. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. *Proceedings of the nineteenth annual conference of the Cognitive Science Societ*y, 107–112.

Fitts, P. M. (1954). The information capacity of the human motot system in controlling amplitude of movement. *Journal of Experimental Psychology, 47*, 381-391.

Hudson, S. E., John, B. E., Knudsen, K., and Byrne, M.D. (1999). A tool for creating predictive performance models from user interface demonstrations. *UIST'99: Proceedings of the ACM Symposium on User Interface Software and Technology, CHI Letters 1*(1), 93-102.

John, B. E. (1994) Toward a deeper comparison of methods: A reaction to Nielsen & Phillips and new data. In *Proceedings Companion of CHI, 1994* (Boston, MA, April 24-28, 1994) ACM, New York, 1994. pp. 285-286.

John, B. E. & Gray, W. D. *GOMS Analyses for Parallel Activities*. Tutorial materials, presented at CHI, 1992 (Monterey, California, May 3- May 7, 1992), CHI, 1994 (Boston MA, April 24-28, 1994) and CHI, 1995 (Denver CO, May 7-11, 1995) ACM, New York.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. To appear at *CHI 2004*.

John, B. E., Remington, R. W., & Steier, D. M. (1991) *An analysis of space shuttle countdown activities: Preliminaries to a computational model of the NASA test director*. Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-91-138.

John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002) Automating CPM-GOMS. *Proceedings of CHI, 2002* (Minneapolis, April 20-25, 2002) ACM, New York.

Kieras, D. E., Meyer, D. E., Ballas, J. A., & Lauber, E. J. (2000). Modern computational perspectives on executive mental processes and cognitive control: Where to from here?. In S. Monsell & J. Driver (Eds.), Control of Cognitive Processes: Attention and Performance XVIII (pp. 681-712). Cambridge, MA: MIT Press.

Koedinger, K. R., Aleven, V., and Heffernan, N. (2003). Toward a rapid development environment for cognitive tutors. *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies, Proceedings of AI-ED 2003* (pp. 455-457). IOS Press.

Kurosu, M. (1994). Dual task model: An evaluation model for the complex operation. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems* (pp. 125-126). New York: ACM Press.

Lallement, Y., & John, B. E. (1998). Cognitive architecture and modeling idiom: An examination of three models of the Wickens' task. Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, August 1998.

Lewis, R. L. (1999). Specifying architectures for language processing: Process, control, and memory in parsing and interpretation. In Crocker, M., Pickering, M., and Clifton, C. Jr., (editors) *Architectures and Mechanisms for Language Processing*. Cambridge University Press.

Nelson, G., Lehman, J. F., John, B. E. (1994a) Experiences in interruptible language processing, In Proceedings of the 1994 AAAI Spring Symposium on Active Natural Language Processing, 1994.

Nelson, G. H., Lehman, J. F., & John, B. E. (1994b) Integrating cognitive capabilities in a real-time task. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, August 1994. pp. 353-358.

Remington, R. John, B. E., Vera, A., Matessa, M., Freed, M., Dalal, M., Harris, R., & Dahlman, E. (2002) *Apex/CPM-GOMS: Modeling human performance in applied HCI domains.* Tutorial materials presented at the Twenty-Fourth Annual Conference of the Cognitive Science Society, (George Mason University, Fairfax VA. August 7, 2002).

Salvucci, D. D. (2001). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies,* 55, 85-107.

Salvucci, D. D., Boer, E. R., & Liu, A. (2001). Toward an integrated model of driver behavior in a cognitive architecture. *Transportation Research Record,* 1779, 9-16.

Salvucci, D. D., John, B. E., Prevas, K., & Centgraf, P. (2004) *Interfaces on the road: Rapid evaluation of in-vehicle devices.* Manuscript in preparation.

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* (pp. 265-272). New York: ACM Press.

Salvucci, D. D., & Macuga, K. L. (2002). Predicting the effects of cellular-phone dialing on driver performance. *Cognitive Systems Research,* 3, 95-102.